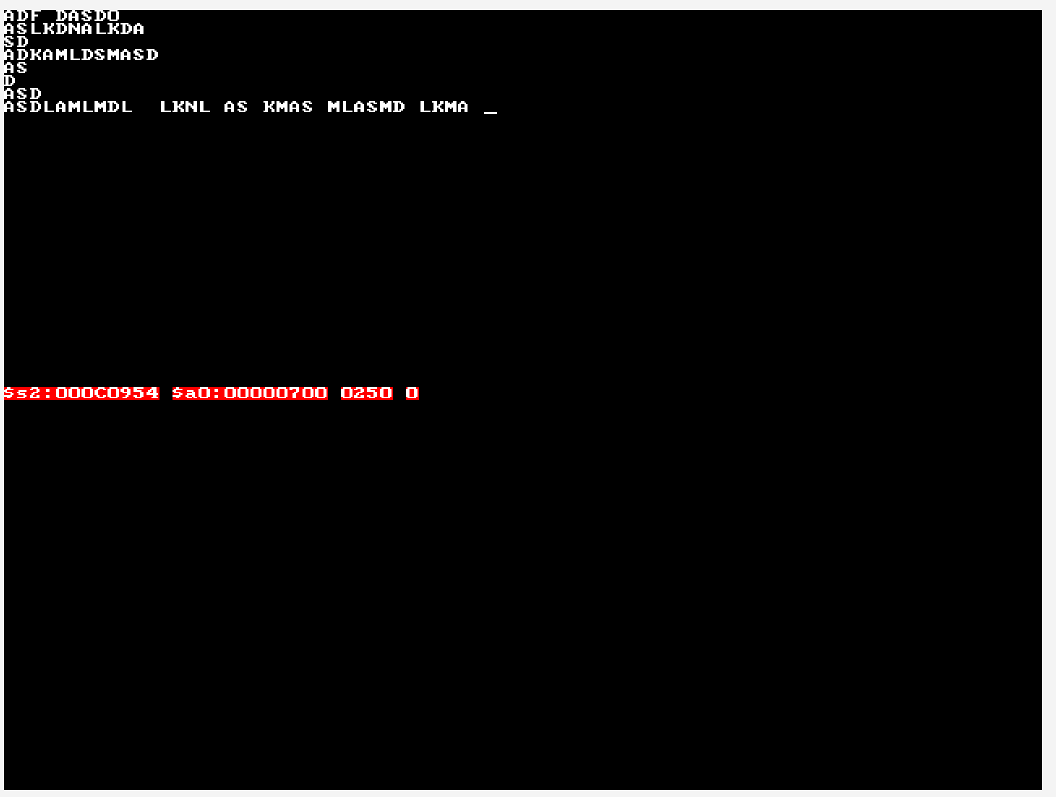
MIPS Assembly Project Report

刘隽良 31380105957

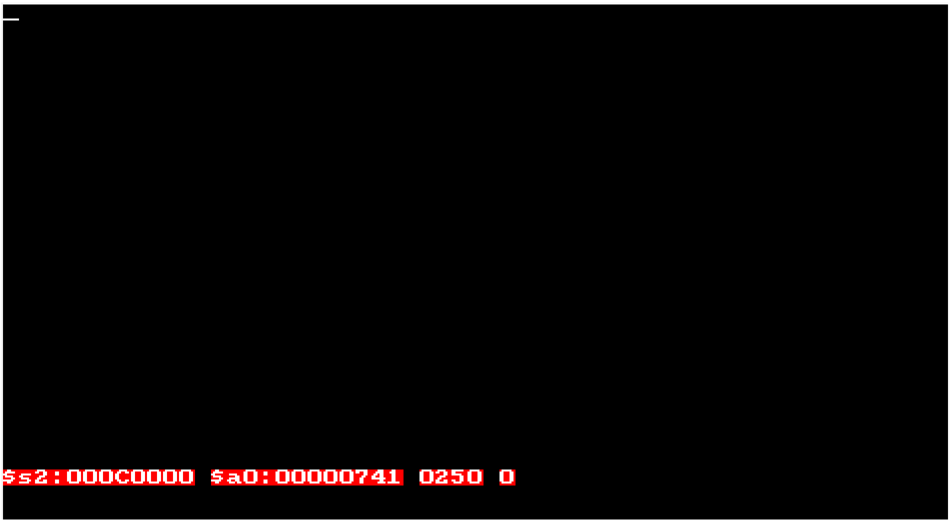
- 1，清屏，esc触发
  - 实现

```
clr1:
 lui $s2, 12
 addi $s2,$s2,0x243c
 j clr
clr:
 sw $zero, 0($s2)
 beq $s2, $zero, main
 addi $s2, $s2, -4
 lui $t4, 12
 slt $t5, $s2, $t4
 bne $t5, $zero, main
 j clr
```

    - 测试结果



```
ADF DASDO
ASLKDNALKDA
SD
ADKAMLDSMASD
AS
D
ASD
ASDLAMLMDL  LKNL AS KMAS MLASMD LKMA _











$s2:000C0954 $a0:00000700 0250 0
```



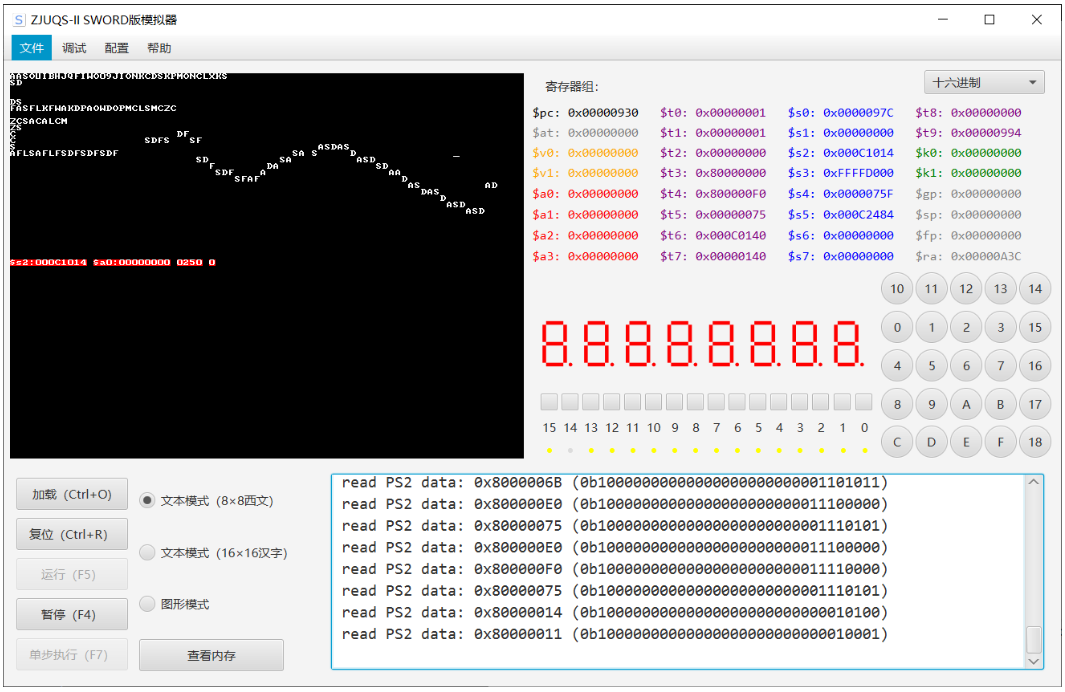
```
$s2:000C0000 $a0:00000741 0250 0
```

2，光标移动

- 实现

```
up:
 lui $t6, 12
 addi $t6, $t6, 320 #end of first line
 slt $t0, $s2, $t6
 bne $t0, $zero, read
 sw $s7, 0($s2)
 addi $t3, $zero, 320
 sub $s2, $s2, $t3
 addi $t3, $zero, 0x075f
 lw $s7, 0($s2)
 sw $t3, 0($s2)
 j read

left:
 sw $s7, 0($s2)
 addi $s2, $s2, -4
 addi $t0, $zero, 0x075f
 lw $s7, 0($s2)
 sw $t0, 0($s2)
 j read

right:
 sw $s7, 0($s2)
 addi $s2, $s2, 4
 addi $t0, $zero, 0x075f
 lw $s7, 0($s2)
 sw $t0, 0($s2)
 j read

down:
 lui $t6, 12
 addi $t6, $t6, 0x22fc #end of first line
 slt $t0, $t6, $s2
 bne $t0, $zero, read
 sw $s7, 0($s2)
 addi $s2, $s2, 320
 addi $t3, $zero, 0x075f
 lw $s7, 0($s2)
 sw $t3, 0($s2)
 j read
```

- 效果



```
AASOUIBHJQFIWOO9JIONKCDSKPMONCLXKS
SD

DS
FASFLKFWAKDPAOWDOPMCLSMCZC
ZCSACALCM
ZS
AFLSAFLFSDFSDFSDF

$s2:000C1014 $a0:00000000 0250 0
```

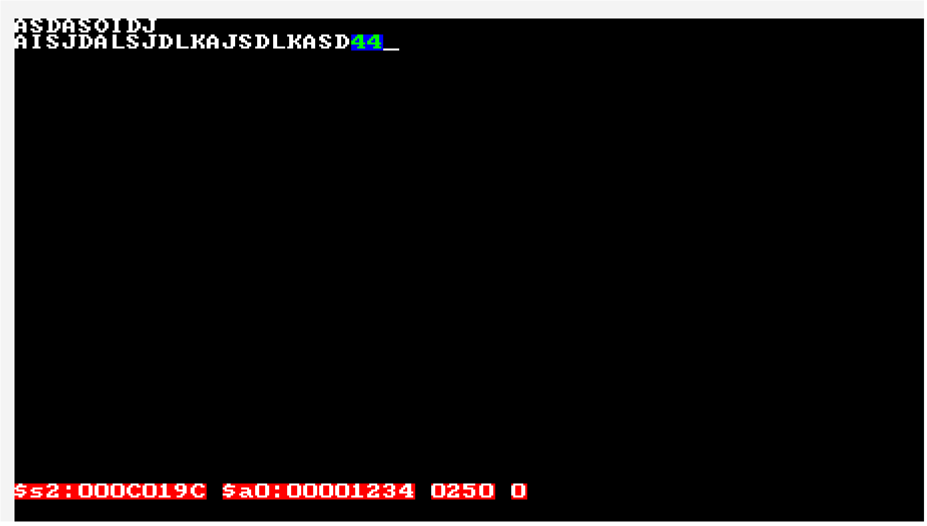
```
$pc: 0x00000930  $t0: 0x00000001  $s0: 0x0000097C  $t8: 0x00000000
$at: 0x00000000  $t1: 0x00000001  $s1: 0x00000000  $t9: 0x00000994
$v0: 0x00000000  $t2: 0x00000000  $s2: 0x000C1014  $k0: 0x00000000
$v1: 0x00000000  $t3: 0x80000000  $s3: 0xFFFFD000  $k1: 0x00000000
$a0: 0x00000000  $t4: 0x800000F0  $s4: 0x0000075F  $gp: 0x00000000
$a1: 0x00000000  $t5: 0x00000075  $s5: 0x000C2484  $sp: 0x00000000
$a2: 0x00000000  $t6: 0x000C0140  $s6: 0x00000000  $fp: 0x00000000
$a3: 0x00000000  $t7: 0x00000140  $s7: 0x00000000  $ra: 0x00000A3C
```

```
read PS2 data: 0x8000006B (0b10000000000000000000000001101011)
read PS2 data: 0x800000E0 (0b10000000000000000000000011100000)
read PS2 data: 0x80000075 (0b10000000000000000000000001110101)
read PS2 data: 0x800000E0 (0b10000000000000000000000011100000)
read PS2 data: 0x800000F0 (0b10000000000000000000000011110000)
read PS2 data: 0x80000075 (0b10000000000000000000000001110101)
read PS2 data: 0x80000014 (0b10000000000000000000000000010100)
read PS2 data: 0x80000011 (0b10000000000000000000000000010001)
```

- 3、在当前光标位置显示字符ASCII码，光标右移，f1触发
  - 实现

```
f1:
 lui $t0, 12
 beq $t0, $s2, read
 addi $t0, $s2, -4
 lw $t0, ($t0)
 andi $s4, $t0, 0x00f0
 srl $s4, $s4, 4
 addi $s4, $s4, 0x30
 jal display
 andi $s4, $t0, 0x000f
 slti $t0, $s4, 0xa
 beq $t0, $zero, char
 addi $s4, $s4, 0x30
 jal display
 j color
```

  - 测试

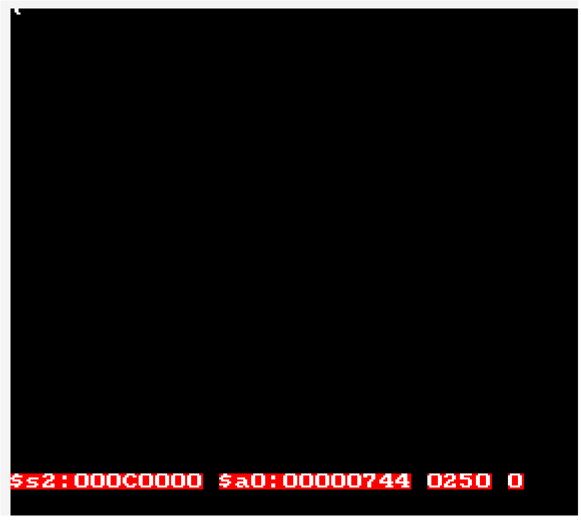


```
ASDASOIDJ
AISJDALSJDLKAJSDLKASD44_

$s2:000C019C $a0:00001234 0250 0
```

- 4, 读取PS2键盘扫描码，存入扫描码缓冲区
  - 实现

```
read:
lw $t1, 0($s3)
lui $t3, 0x8000
and $t2, $t1, $t3
beq $t2, $zero, read_r #check if read
andi $t5, $t1, 0xffff
addi $s4, $zero, 0x41
jal display
j read
```

  - 测试



```
$s2:000C0000 $a0:00000744 0250 0
```

- 5, 从缓冲区读取扫描码转换为ASCII码，存入字符缓冲区，并在当前光标处显示
  - 实现

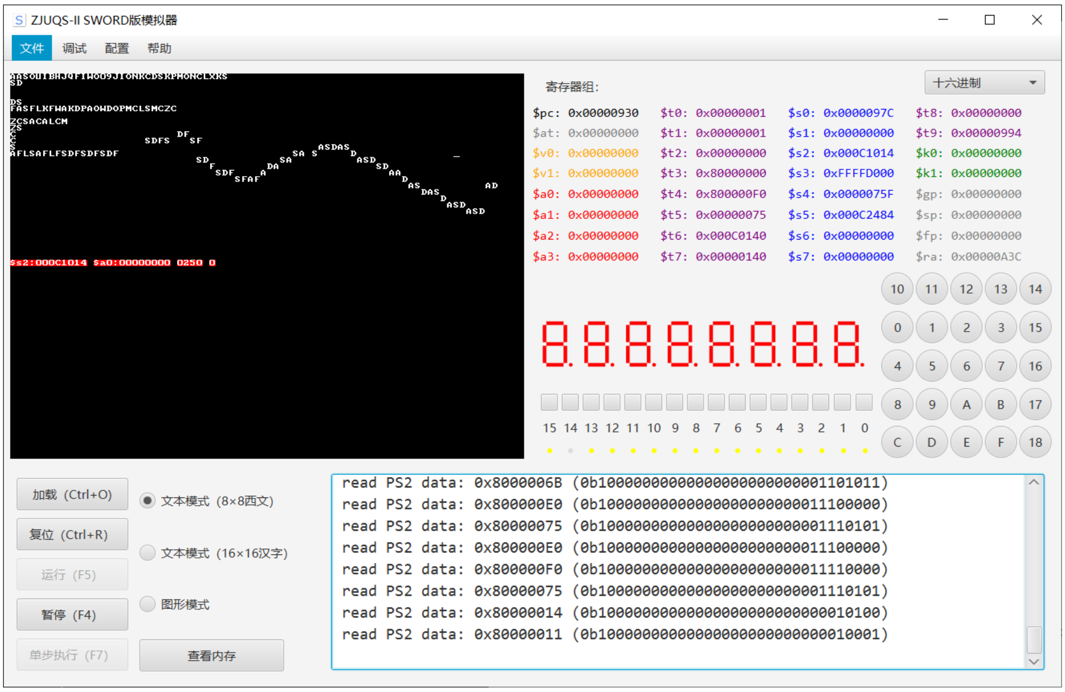
```
read_r:
lw $t1, 0($s3)
lui $t3, 0x8000
and $t2, $t1, $t3
beq $t2, $zero, read_r #check if read
andi $t5, $t1, 0xffff
 addi $t4, $zero, 0x75 #up
 beq $t5, $t4, up
 addi $t4, $zero, 0x6B
```

```
beq $t5, $t4, left
addi $t4, $zero, 0x72
beq $t5, $t4, down
addi $t4, $zero, 0x74
beq $t5, $t4, right
add $s7, $zero, $zero
addi $t4, $zero, 0x1c #a
beq $t5, $t4, a
addi $t4, $zero, 0x32 #b
beq $t5, $t4, b
addi $t4, $zero, 0x21 #c
beq $t5, $t4, c
addi $t4, $zero, 0x23 #d
beq $t5, $t4, d
addi $t4, $zero, 0x24 #e
beq $t5, $t4, e
addi $t4, $zero, 0x2b #f
beq $t5, $t4, f
addi $t4, $zero, 0x34 #g
beq $t5, $t4, g
addi $t4, $zero, 0x33 #h
beq $t5, $t4, h
addi $t4, $zero, 0x43 #i
beq $t5, $t4, i
addi $t4, $zero, 0x3b #j
beq $t5, $t4, j
addi $t4, $zero, 0x42 #k
beq $t5, $t4, k
addi $t4, $zero, 0x4b #l
beq $t5, $t4, l
addi $t4, $zero, 0x3a #m
beq $t5, $t4, m
addi $t4, $zero, 0x31 #n
beq $t5, $t4, n
addi $t4, $zero, 0x44 #o
beq $t5, $t4, o
addi $t4, $zero, 0x4d #p
beq $t5, $t4, p
addi $t4, $zero, 0x15 #q
beq $t5, $t4, q
addi $t4, $zero, 0x2d #r
beq $t5, $t4, r
addi $t4, $zero, 0x1b #s
beq $t5, $t4, s
addi $t4, $zero, 0x2c #t
beq $t5, $t4, t
addi $t4, $zero, 0x3c #u
beq $t5, $t4, u
addi $t4, $zero, 0x2a #v
beq $t5, $t4, v
addi $t4, $zero, 0x1d #w
beq $t5, $t4, w
addi $t4, $zero, 0x22 #x
beq $t5, $t4, x
addi $t4, $zero, 0x35 #y
beq $t5, $t4, y
addi $t4, $zero, 0x1a #z
beq $t5, $t4, z
```

```
 addi $t4, $zero, 0x16 #1
 beq $t5, $t4, n1
 addi $t4, $zero, 0x1e #2
 beq $t5, $t4, n2
 addi $t4, $zero, 0x26 #3
 beq $t5, $t4, n3
 addi $t4, $zero, 0x25 #4
 beq $t5, $t4, n4
 addi $t4, $zero, 0x2e #5
 beq $t5, $t4, n5
 addi $t4, $zero, 0x36 #6
 beq $t5, $t4, n6
 addi $t4, $zero, 0x3d #7
 beq $t5, $t4, n7
 addi $t4, $zero, 0x3e #8
 beq $t5, $t4, n8
 addi $t4, $zero, 0x46 #9
 beq $t5, $t4, n9
 addi $t4, $zero, 0x45 #0
 beq $t5, $t4, n0
 addi $t4, $zero, 0x29 #space
 beq $t5, $t4, space
 addi $t4, $zero, 0x5a #enter
 beq $t5, $t4, enter
 addi $t4, $zero, 0x66 #backspace
 beq $t5, $t4, back_space
 addi $t4, $zero, 0x76
 beq $t5, $t4, clr1 #clear screen
 addi $t4, $zero, 0x05
 beq $t5, $t4, f1 #f1
 addi $t4, $zero, 0x06
 beq $t5, $t4, f2 #f2
 addi $t4, $zero, 0x04
 beq $t5, $t4, f3 #f3
 j read

a:
 addi $s4, $zero, 0x41
 jal display
 j read
b:
 addi $s4, $zero, 0x42
 jal display
 j read
c:
 addi $s4, $zero, 0x43
 jal display
 j read
d:
 addi $s4, $zero, 0x44
 jal display
 j read
e:
 addi $s4, $zero, 0x45
 jal display
 j read
f:
 addi $s4, $zero, 0x46
 jal display
 j read
g:
 addi $s4, $zero, 0x47
 jal display
 j read
h:
 addi $s4, $zero, 0x48
 jal display
 j read
i:
 addi $s4, $zero, 0x49
 jal display
 j read
j:
 addi $s4, $zero, 0x4a
 jal display
 j read
k:
 addi $s4, $zero, 0x4b
 jal display
 j read
l:
 addi $s4, $zero, 0x4c
 jal display
 j read
m:
 addi $s4, $zero, 0x4d
 jal display
 j read
n:
 addi $s4, $zero, 0x4e
 jal display
 j read
o:
 addi $s4, $zero, 0x4f
 jal display
 j read
p:
 addi $s4, $zero, 0x50
 jal display
 j read
q:
 addi $s4, $zero, 0x51
 jal display
 j read
r:
 addi $s4, $zero, 0x52
 jal display
 j read
s:
 addi $s4, $zero, 0x53
 jal display
 j read
t:
 addi $s4, $zero, 0x54
 jal display
 j read
```

```
u:
 addi $s4, $zero, 0x55
 jal display
 j read
v:
 addi $s4, $zero, 0x56
 jal display
 j read
w:
 addi $s4, $zero, 0x57
 jal display
 j read
x:
 addi $s4, $zero, 0x58
 jal display
 j read
y:
 addi $s4, $zero, 0x59
 jal display
 j read
z:
 addi $s4, $zero, 0x5a
 jal display
 j read
n0:
 addi $s4, $zero, 0x30
 jal display
 j read
n1:
 addi $s4, $zero, 0x31
 jal display
 j read
n2:
 addi $s4, $zero, 0x32
 jal display
 j read
n3:
 addi $s4, $zero, 0x33
 jal display
 j read
n4:
 addi $s4, $zero, 0x34
 jal display
 j read
n5:
 addi $s4, $zero, 0x35
 jal display
 j read
n6:
 addi $s4, $zero, 0x36
 jal display
 j read
n7:
 addi $s4, $zero, 0x37
 jal display
 j read
n8:
 addi $s4, $zero, 0x38
 jal display
 j read
n9:
 addi $s4, $zero, 0x39
 jal display
 j read
space:
 add $s4, $zero, $zero
 jal display
 j read
```

  - 测试



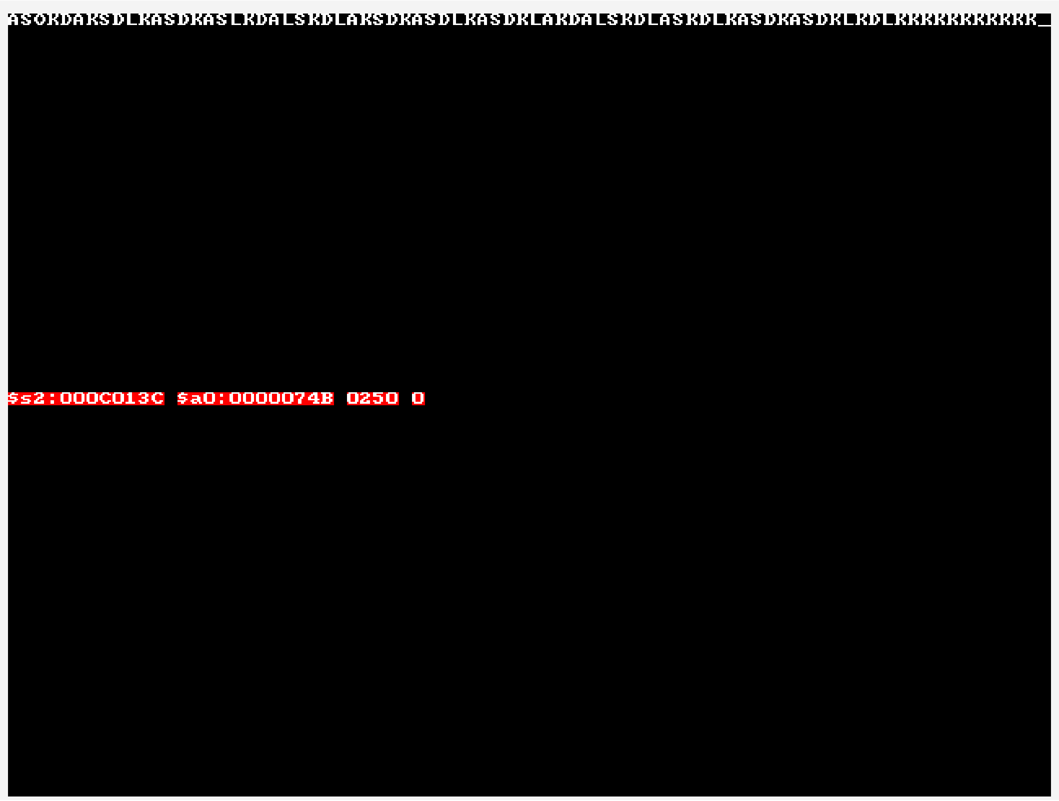
```
$pc: 0x00000930   $t0: 0x00000001   $s0: 0x0000097C   $t8: 0x00000000
$at: 0x00000000   $t1: 0x00000001   $s1: 0x00000000   $t9: 0x00000994
$v0: 0x00000000   $t2: 0x00000000   $s2: 0x000C1014   $k0: 0x00000000
$v1: 0x00000000   $t3: 0x80000000   $s3: 0xFFFFD000   $k1: 0x00000000
$a0: 0x00000000   $t4: 0x800000F0   $s4: 0x0000075F   $gp: 0x00000000
$a1: 0x00000000   $t5: 0x00000075   $s5: 0x000C2484   $sp: 0x00000000
$a2: 0x00000000   $t6: 0x000C0140   $s6: 0x00000000   $fp: 0x00000000
$a3: 0x00000000   $t7: 0x00000140   $s7: 0x00000000   $ra: 0x00000A3C
```

```
read PS2 data: 0x8000006B (0b10000000000000000000000001101011)
read PS2 data: 0x800000E0 (0b10000000000000000000000011100000)
read PS2 data: 0x80000075 (0b10000000000000000000000001110101)
read PS2 data: 0x800000E0 (0b10000000000000000000000011100000)
read PS2 data: 0x800000F0 (0b10000000000000000000000011110000)
read PS2 data: 0x80000075 (0b10000000000000000000000001110101)
read PS2 data: 0x80000014 (0b10000000000000000000000000010100)
read PS2 data: 0x80000011 (0b10000000000000000000000000010001)
```

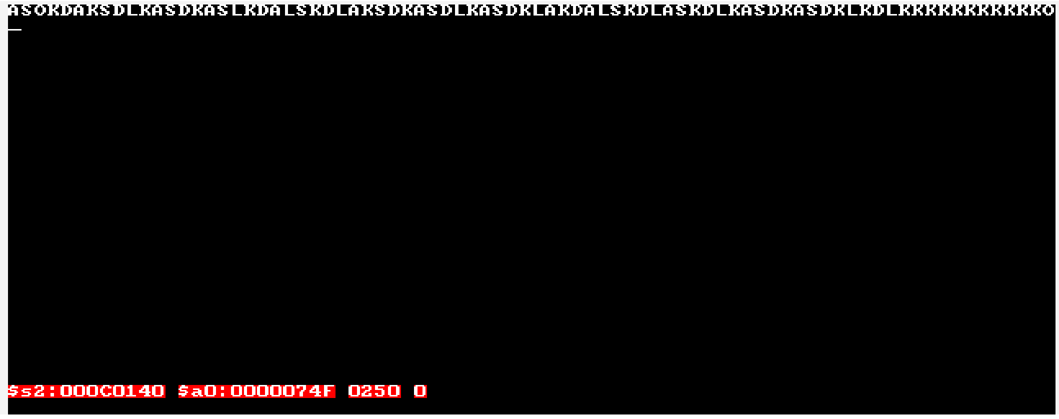
- 6，在当前光标位置显示一个字符，光标右移一列，判断边界
  - 实现

```
同上
```

  - 测试



```
ASOKDAKSDLKASDKASLKDALSKDLAKSDKASDLKASDKLAKDALSKDLASKDLKASDKASDKLKDLKKKKKKKKKKK_

$s2:000C013C $a0:0000074B 0250 0
```

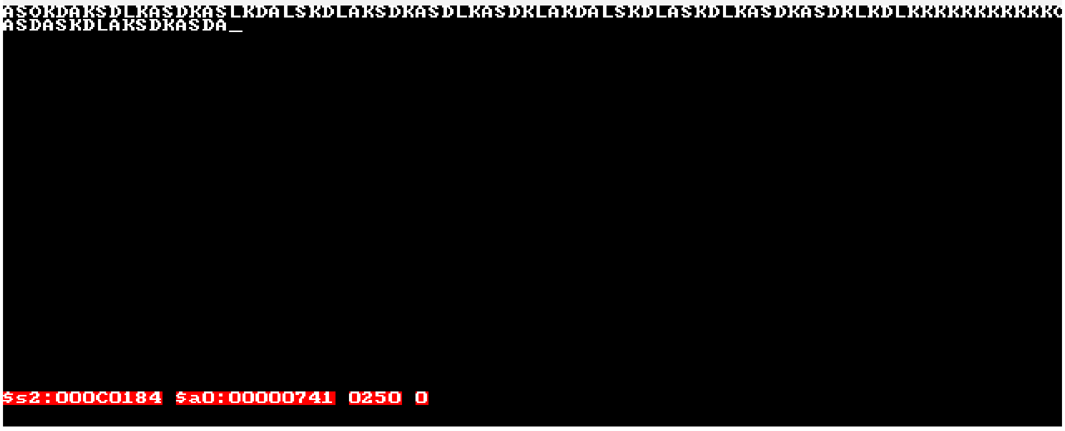


```
ASOKDAKSDLKASDKASLKDALSKDLAKSDKASDLKASDKLAKDALSKDLASKDLKASDKASDKLKDLKKKKKKKKKKKO
_

$s2:000C0140 $a0:0000074F 0250 0
```

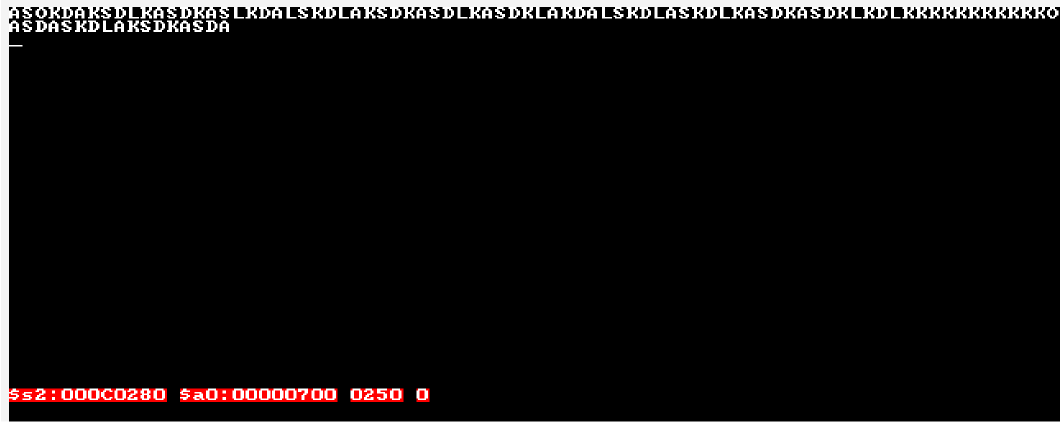
- 7. 换行，光标置下一行首
  - 实现

```
enter:
 lui $t0, 12
 addi $t0, $t0, 0x2300
 slt $t1, $s2, $t0
 beq $t1, $zero, read
 add $s4, $zero, $zero
 jal display
 lui $t6, 12
 sub $t6, $s2, $t6
 addi $t7, $zero, 320
 Loop2:
 sub $t6, $t6, $t7
 beq $t6, $zero, read
 slt $t8, $t6, $zero # $t8 = $t6<0?1:0
 bne $t8, $zero, enter
 j Loop2
```

  - 测试



```
ASOKDAKSDLKASDKASLKDALSKDLAKSDKASDLKASDKLAKDALSKDLASKDLKASDKASDKLKDLKKKKKKKKKKKO
ASDASKDLAKSDKASDA_









$s2:000C0184 $a0:00000741 0250 0
```

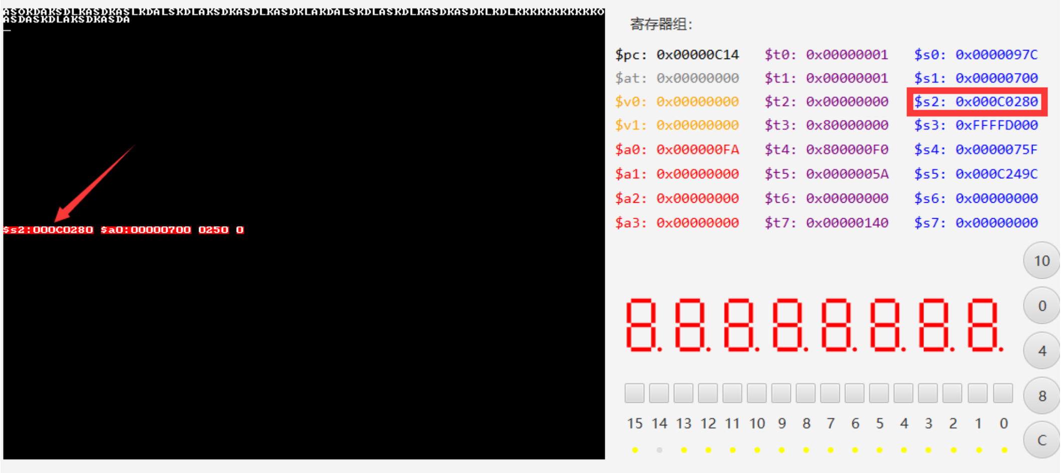


```
ASOKDAKSDLKASDKASLKDALSKDLAKSDKASDLKASDKLAKDALSKDLASKDLKASDKASDKLKDLKKKKKKKKKKKO
ASDASKDLAKSDKASDA
_









$s2:000C0280 $a0:00000700 0250 0
```

- 8,在屏幕指定位置显示当前寄存器值
    - 实现

```
lui $s5, 12
 addi $s5, $s5, 0x2440
 addi $s6, $zero, 0x4724 #$
 jal dispc
 addi $s6, $zero, 0x4773 #s
 jal dispc
 addi $s6, $zero, 0x4732 #2
 jal dispc
 addi $s6, $zero, 0x473A #:
 jal dispc
 srl $s6, $s2, 28  #s2 8
 jal process
 jal dispc
 srl $s6, $s2, 24
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s2, 20
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s2, 16
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s2, 12
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s2, 8
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s2, 4
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 addi $s6, $s2, 0
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
```

- 测试



```
ASOKDAKSDLKASDKASLKDALSKDLAKSDKASDLKASDKLAKDALSKDLASKDLKASDKASDKLKDLKKKKKKKKKKKO
ASDASKDLAKSDKASDA
_
```

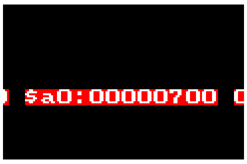
```
$s2:000C0280 $a0:00000700 0250 0
```

寄存器组:

```
$pc: 0x00000C14   $t0: 0x00000001   $s0: 0x0000097C
$at: 0x00000000   $t1: 0x00000001   $s1: 0x00000700
$v0: 0x00000000   $t2: 0x00000000   $s2: 0x000C0280
$v1: 0x00000000   $t3: 0x80000000   $s3: 0xFFFFD000
$a0: 0x000000FA   $t4: 0x800000F0   $s4: 0x0000075F
$a1: 0x00000000   $t5: 0x0000005A   $s5: 0x000C249C
$a2: 0x00000000   $t6: 0x00000000   $s6: 0x00000000
$a3: 0x00000000   $t7: 0x00000140   $s7: 0x00000000
```

- 10.在屏幕指定位置，用16进制显示指定内存单元数据。地址参数：$a0
  - 实现

```
showdata:
 lw $s1, -4($a0)
 add $s0, $zero, $ra
 addi $s6, $zero, 0x4724 #$
 jal dispc
 addi $s6, $zero, 0x4761 #a
 jal dispc
 addi $s6, $zero, 0x4730 #0
 jal dispc
 addi $s6, $zero, 0x473A #:
 jal dispc
 srl $s6, $s1, 28  #s2 8
 jal process
 jal dispc
 srl $s6, $s1, 24
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s1, 20
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s1, 16
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s1, 12
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s1, 8
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 srl $s6, $s1, 4
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 addi $s6, $s1, 0
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4700
 jal process
 jal dispc
 jr $s0
```

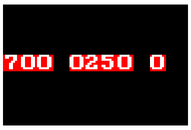
  - 测试
  - 

```
$a0:00000700
```

- 10.,二进制转换为十进制并在屏幕指定位置显示
  - 实现

```
showbin:
 addi $t9, $ra, 0
 add $t1, $zero, $zero
 loopd3:
 slti $t2, $a0, 1000
 bne $t2, $zero, loopd2
 addi $a0, $a0, -1000
 addi $t1, $t1, 0x1000
 j loopd3
 loopd2:
 slti $t2, $a0, 100
 bne $t2, $zero, loopd1
 addi $a0, $a0, -100
 addi $t1, $t1, 0x100
 j loopd2
 loopd1:
 slti $t2, $a0, 10
 bne $t2, $zero, loopd0
 addi $a0, $a0, -10
 addi $t1, $t1, 0x10
 j loopd1
 loopd0:
 slti $t2, $a0, 1
 bne $t2, $zero, showbin2
 addi $a0, $a0, -1
 addi $t1, $t1, 0x1
 j loopd0
 showbin2:
 srl $s6, $t1, 12
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4730
 jal dispc
 srl $s6, $t1, 8
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4730
 jal dispc
 srl $s6, $t1, 4
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4730
 jal dispc
 addi $s6, $t1, 0
 andi $s6, $s6, 0xf
 addi $s6, $s6, 0x4730
 jal dispc
 jr $t9
```

  - 测试



```
700 0250 0
```

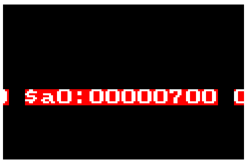
- 11, 16进制数转换成ASCII码并在屏幕指定位置显示
  - 实现

    已经在寄存器数据显示中实现

    相关辅助代码

```
process:
 andi $s6, $s6, 0xf
 slti $t1, $s6, 0xa
 bne $t1, $zero, num
 addi $s6, $s6, -0xa
 addi $s6, $s6, 0x4741
 jr $ra
num:
 addi $s6, $s6, 0x4730
 jr $ra
```

  - 测试

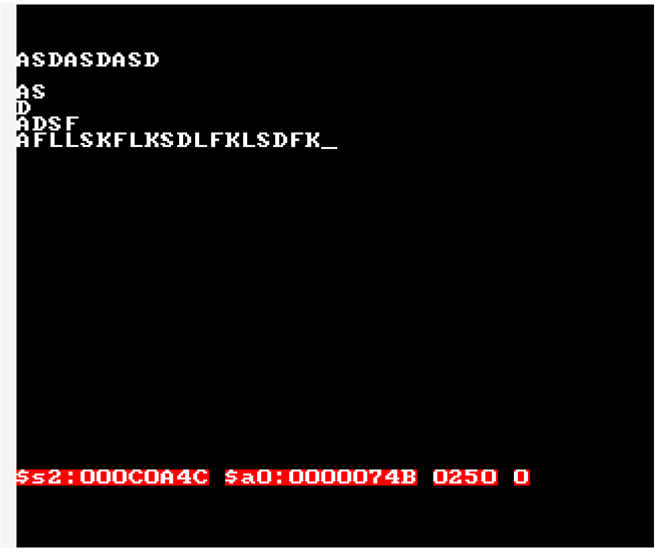


```
$a0:00000700
```

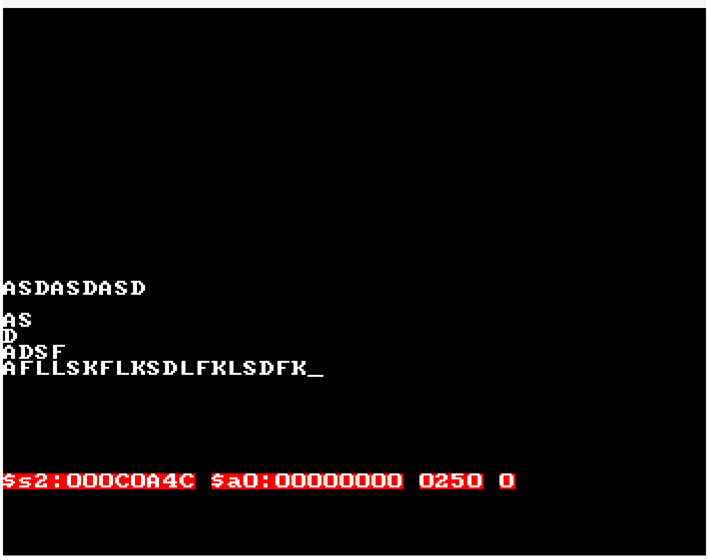
- 12.屏幕上滚或下滚一行，f2 f3触发
  - 实现

```
f2: #scroll up
 lui $s5, 12
 lui $s6, 12
 addi $s6, $s6, 0x2300
 loop3:
 beq $s5, $s6, lastline
 lw $t0, 0x140($s5)
 sw $t0, 0($s5)
 addi $s5, $s5, 4
 j loop3
 lastline:
 lui $s6, 12
 addi $s6, $s6, 0x2440
 loop4:
 beq $s5, $s6, read
 sw $zero, 0($s5)
 addi $s5, $s5, 4
 j loop4

f3:
 lui $s5, 12
 addi $s5, $s5, 0x243c
 lui $s6, 12
 addi $s6, $s6, 0x13c
 loop5:
 beq $s5, $s6, firstline
 lw $t0, -0x140($s5)
 sw $t0, 0($s5)
 addi $s5, $s5, -4
 j loop5
 firstline:
 lui $s6, 12
 addi $s6, $s6, -4
 loop6:
 beq $s5, $s6, read
 sw $zero, 0($s5)
 addi $s5, $s5, -4
 j loop6
```

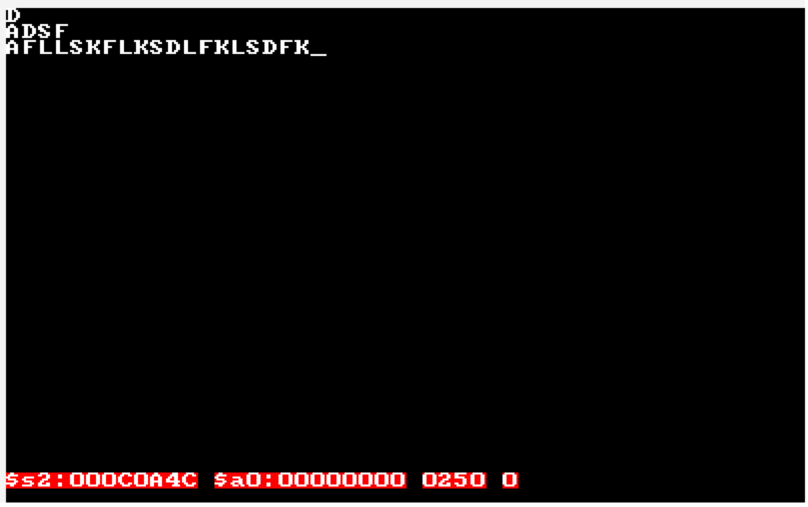
  - 测试



```
ASDASDASD
AS
D
ADSF
AFLLSKFLKSDLFKLSDFK_


$s2:000C0A4C $a0:0000074B 0250 0
```



```
ASDASDASD
AS
D
ADSF
AFLLSKFLKSDLFKLSDFK_


$s2:000C0A4C $a0:00000000 0250 0
```



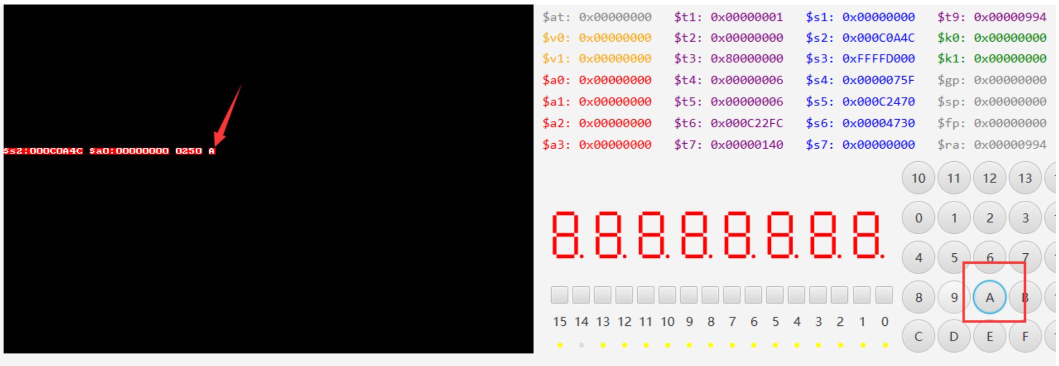
```
D
ADSF
AFLLSKFLKSDLFKLSDFK_


$s2:000C0A4C $a0:00000000 0250 0
```

- 13读取阵列键盘16进制数并在光标处显示
  - 实现

```
showbutton:
 lui $t1, 0xffff
 addi $t1, $zero, 0xfc00
 lw $t0, 0($t1)
 andi $s6, $t0, 0x000f
 slti $t0, $s6, 0xa
 bne $t0, $zero, sb1
 addi $s6, $s6, -0xa
```

```
  addi $s6, $s6, 0x4741
  sw  $s6, 0($s5) # to display
  addi $s5, $s5, 4
  jr $ra
sb1:
  addi $s6, $s6, 0x4730
  sw  $s6, 0($s5) # to display
  addi $s5, $s5, 4
  jr $ra
```

- 测试

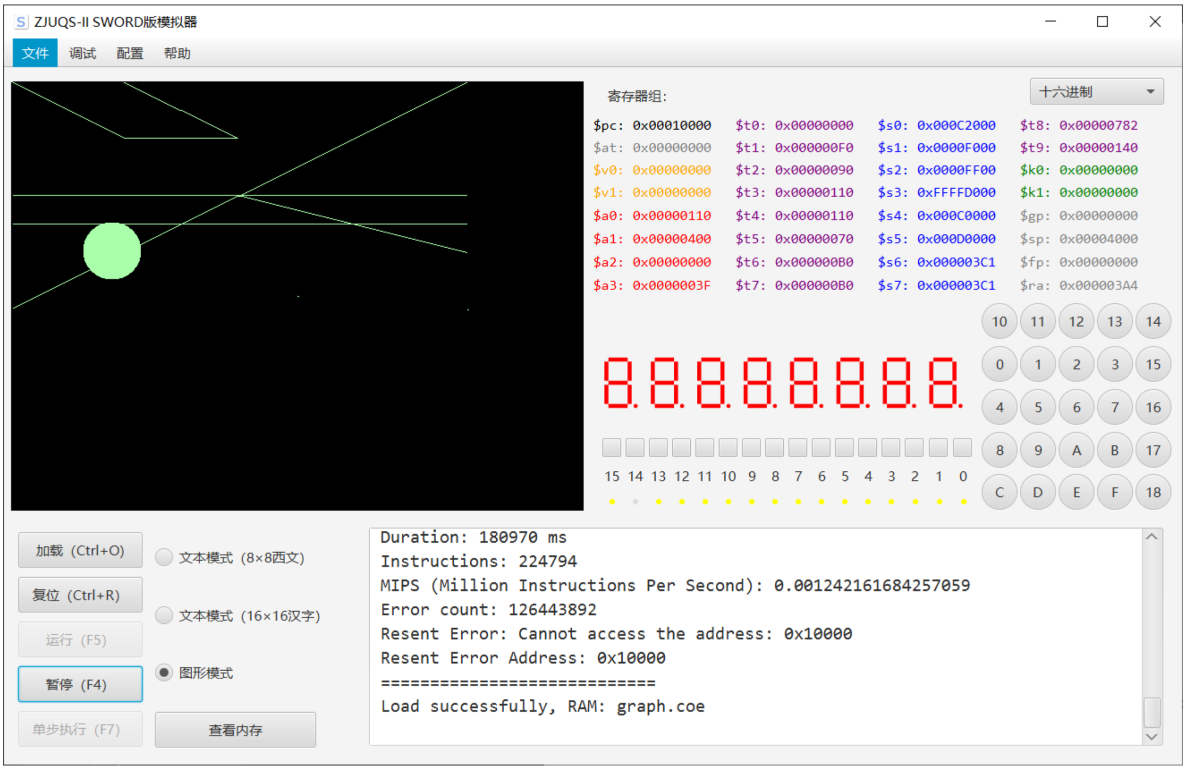


```
$s2:000C0A4C $a0:00000000 0250 A
```

```
$at: 0x00000000   $t1: 0x00000001   $s1: 0x00000000   $t9: 0x00000994
$v0: 0x00000000   $t2: 0x00000000   $s2: 0x000C0A4C   $k0: 0x00000000
$v1: 0x00000000   $t3: 0x80000000   $s3: 0xFFFFD000   $k1: 0x00000000
$a0: 0x00000000   $t4: 0x00000006   $s4: 0x0000075F   $gp: 0x00000000
$a1: 0x00000000   $t5: 0x00000006   $s5: 0x000C2470   $sp: 0x00000000
$a2: 0x00000000   $t6: 0x000C22FC   $s6: 0x00004730   $fp: 0x00000000
$a3: 0x00000000   $t7: 0x00000140   $s7: 0x00000000   $ra: 0x00000994
```

基本作业B

提交代码效果总览



寄存器组：

```
$pc: 0x00010000   $t0: 0x00000000   $s0: 0x000C2000   $t8: 0x00000782
$at: 0x00000000   $t1: 0x000000F0   $s1: 0x0000F000   $t9: 0x00000140
$v0: 0x00000000   $t2: 0x00000090   $s2: 0x0000FF00   $k0: 0x00000000
$v1: 0x00000000   $t3: 0x00000110   $s3: 0xFFFFD000   $k1: 0x00000000
$a0: 0x00000110   $t4: 0x00000110   $s4: 0x000C0000   $gp: 0x00000000
$a1: 0x00000400   $t5: 0x00000070   $s5: 0x000D0000   $sp: 0x00004000
$a2: 0x00000000   $t6: 0x000000B0   $s6: 0x000003C1   $fp: 0x00000000
$a3: 0x0000003F   $t7: 0x000000B0   $s7: 0x000003C1   $ra: 0x000003A4
```

○ 文本模式（8×8西文）
○ 文本模式（16×16汉字）
● 图形模式

```
Duration: 180970 ms
Instructions: 224794
MIPS (Million Instructions Per Second): 0.001242161684257059
Error count: 126443892
Resent Error: Cannot access the address: 0x10000
Resent Error Address: 0x10000
===========================
Load successfully, RAM: graph.coe
```

- 1，指定位置显示点
  - 实现

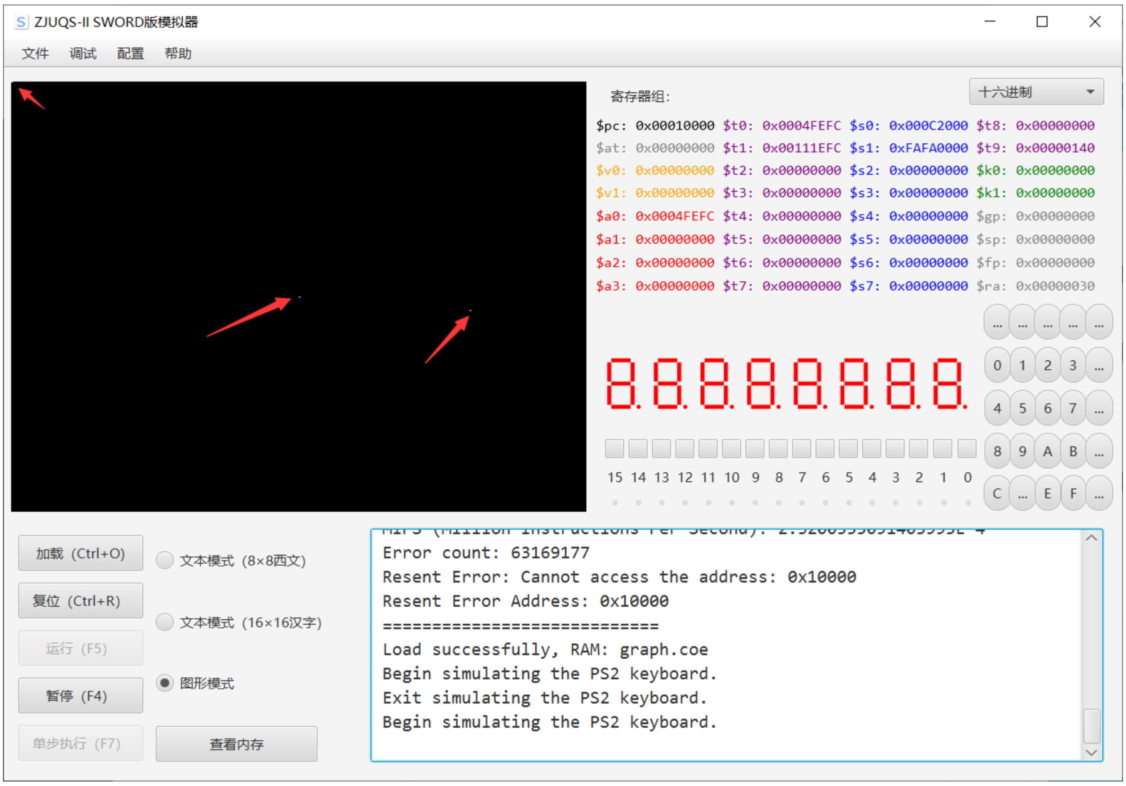
```
pointad: #pointad($a0) draw at xxyy
 andi $t0, $a0, 0xff #get line
 srl $a0, $a0, 8
 andi $a0, $a0, 0xff #get column
 addi $t9, $zero, 0x140
 mult $t0, $t9
 mflo $t0 #line * 320
 add $t0, $t0, $a0 #line * 320 + column
 sll $t0, $t0, 2 # pixel * 4
 add $a0, $zero, $t0
 j point
point: #point(#a0) draw at $a0
 add $t1, $s0, $a0
 sw $s1, 0($t1)
 jr $ra
```

可根据a0中地址计算实际位置并画点

- 测试
  - 测试代码

```
  main:
   lui $s1, 0xfafa
   addi $s1, $s1, 0x0 #white
   lui $s0, 12
   addi $s0, $s0, 0x2000 #s0 <- first row
   j test
  test:
   add $a0, $zero, $zero #first pixel
   jal pointad
   addi $a0, $zero, 0xA0F0 #in the middle xx: 160, tt: 240
   jal pointad
 addi $a0, $zero, 0xffff #in the max xx: 255 yy: 255
   jal pointad
 j end
```

- 测试结果



```
$pc: 0x00010000 $t0: 0x0004FEFC $s0: 0x000C2000 $t8: 0x00000000
$at: 0x00000000 $t1: 0x00111EFC $s1: 0xFAFA0000 $t9: 0x00000140
$v0: 0x00000000 $t2: 0x00000000 $s2: 0x00000000 $k0: 0x00000000
$v1: 0x00000000 $t3: 0x00000000 $s3: 0x00000000 $k1: 0x00000000
$a0: 0x0004FEFC $t4: 0x00000000 $s4: 0x00000000 $gp: 0x00000000
$a1: 0x00000000 $t5: 0x00000000 $s5: 0x00000000 $sp: 0x00000000
$a2: 0x00000000 $t6: 0x00000000 $s6: 0x00000000 $fp: 0x00000000
$a3: 0x00000000 $t7: 0x00000000 $s7: 0x00000000 $ra: 0x00000030
```

```
MIPS (Million Instructions Per Second): 2.[illegible]E-4
Error count: 63169177
Resent Error: Cannot access the address: 0x10000
Resent Error Address: 0x10000
===========================
Load successfully, RAM: graph.coe
Begin simulating the PS2 keyboard.
Exit simulating the PS2 keyboard.
Begin simulating the PS2 keyboard.
```

- 2，画直线

- 实现

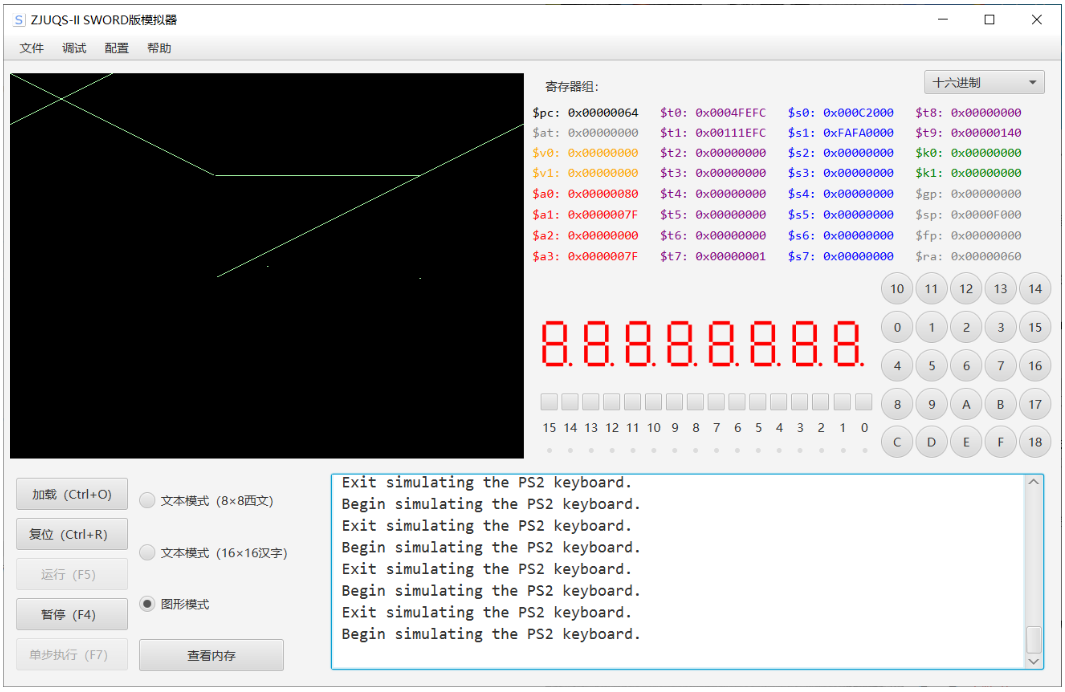
```
drawline:
 addiu $sp, $sp, -56    # Decrement the stack pointer
sw $ra, 48($sp)     # Save the value of the return address ($ra) to the stack
    sw $s0, 44($sp)     # Save the original value of $s0 to the stack
sw $s1, 40($sp)     # Save the original value of $s1 to the stack
    sw $s2, 36($sp)     # Save the original value of $s2 to the stack
sw $s3, 32($sp)     # Save the original value of $s3 to the stack
    sw $s4, 28($sp)     # Save the original value of $s4 to the stack
    sw $s5, 24($sp)     # Save the original value of $s5 to the stack
    sw $s6, 20($sp)     # Save the original value of $s6 to the stack
sw $s7, 16($sp)     # Save the original value of $s7 to the stack
    sw $t0, 12($sp)     # Save the original value of $t0 to the stack
    sw $t1, 8($sp)      # Save the original value of $t1 to the stack
    sw $t2, 4($sp)      # Save the original value of $t2 to the stack
    sw $t3, ($sp)       # Save the original value of $t3 to the stack

    # Copies the values of the parameters passed in into the now free s registers
    srl $s0, $a0, 24
    srl $s1, $a0, 16
    srl $s2, $a0, 8
    move $s3, $a0

    andi $a0, $s0, 0xff
    andi $a1, $s1, 0xff
    andi $a2, $s2, 0xff
    andi $a3, $s3, 0xff

    move $s0, $a0       # Store the x co-ordinate of point 1 in $s0
    move $s1, $a1       # Store the y co-ordinate of point 1 in $s1
    move $s2, $a2       # Store the x co-ordinate of point 2 in $s2
    move $s3, $a3       # Store the y co-ordinate of point 2 in $s3
    lw $s4, 52($sp)     # Store the colour (takes form the stack) in $s4


    # Main code for drawing the line on the bitmap

    subu $t3, $s2, $s0      # Calculates x1 - x0 and store it in $t3
    #abs $s5, $t3           # Sets dx ($s5) to the absolute value of x1 - x0
    move $s5, $s3
    sra $t7,$s5,31
    xor $s5,$s5,$t7
    sub $s5,$s5,$t7

    subu $t3, $s3, $s1      # Calculates y1 - y0 and stores it in $t3
    #abs $s6, $t3           # Sets dy ($s6) to the absolute value of y1 - y0
    move $s6, $t3
    sra $t7,$s6,31
    xor $s6,$s6,$t7
    sub $s6,$s6,$t7
    sub $s6, $zero, $s6     # Sets dy to -dy as dy is needed as a minus
value later in calculations
                            # it is turned minus via two's complement
method

    #bgt $s0, $s2, sxelse   # If x0 is greater than x1 then branch to
sxelse
    slt $t7, $s2, $s0
    bne $t7, $zero, sxelse
    nop
    addi $s7, $zero, 1              # Set the value of sx ($s7) to 1
    b sxcomplete            # Branch around the sxelse section
    nop

sxelse:             # Branches to here if x0 is greater than x1
    addi $s7, $zero, -1     # Sets the value of sx ($s7) to -1

sxcomplete:                 # Branches to here if x1 was greater than
x0
    #bgt $s1, $s3, syelse   # If y0 is greater than y1 then branch to
syelse
    slt $t7, $s3, $s1
    bne $t7, $zero, syelse
    nop
    addi $t0, $zero, 1              # Sets the value of sy ($t0) to 1
    b sycomplete            # Branch around the syelse section
    nop

syelse:             # Branche shere is y0 is greater than y1
    addi $t0, $zero, -1     # Sets the value sy ($t0) to -1

sycomplete:             # Branches to here if y1 was greater than y0
    addu $t1, $s5, $s6  # err is set to the value of dx - dy

drawpixelloop:
    slti $t7, $s0, 0xff
    beq $t7, $zero, exit
    slti $t7, $s1, 0xff
    beq $t7, $zero, exit
    move $a0, $s0       # Store the x co-ordinate in $a0
    move $a1, $s1       # Store the y co-ordinate in $a1
    move $a2, $s4       # Store the colour in $a2

    jal setpixel        # Enter the subroutine "setpixel"

    nop

    add $t2, $t1, $t1           # Sets e2 ($t2) to err ($t1) * 2 by
calculating err + err
    #bgt $s6, $t2, e2notgreaterthandy   # Branch if -dy is greater than
e2
    slt $t7, $t2, $s6
    bne $t7, $zero, e2notgreaterthandy
    nop
    add $t1, $t1, $s6           # Calculate err = err - dy: err = $t1 ,
-dy = $s6
    add $s0, $s0, $s7           # Calculate x0 = x0 + sx: x0 = $s0 , sx
= $s7
```

```
e2notgreaterthandy:
    #bgt $t2, $s5, e2greaterthandx      # Branch if e2 ($t2) is greater than dx ($s5)
    slt $t7, $s5, $t2
    bne $t7, $zero, e2greaterthandx
    nop
    add $t1, $t1, $s5           # Calculate err = derr + dx: err = $t1 , dx = $s5
    add $s1, $s1, $t0           # Caluclate y0 = y0 + sy: y0 = $s1 , sy = $t0
e2greaterthandx:

    # To exit the loop x0 must now be equal to x1 and y0 much now be equal to y1
    # The two statements must be true to pass both branches and thus exits the loop

    bne $s0, $s2, drawpixelloop # If x0 ($s0) and x1 ($s2) are not
    nop             # equal branch to drawpixelloop
    bne $s1, $s3, drawpixelloop # If y0 ($s1) and y1 ($s3) are not
    nop             # equal branch to drawpixelloop

    # Restores the original values of the s registers from the stack
exit:
    lw $t3, ($sp)       # Restore the original value of $t3 from the stack
    lw $t2, 4($sp)      # Restore the original value of $t2 from the stack
    lw $t1, 8($sp)      # Restore the original value of $t1 from the stack
    lw $t0, 12($sp)     # Restore the original value of $t0 from the stack
    lw $s7, 16($sp)     # Restore the original value of $s7 from the stack
    lw $s6, 20($sp)     # Restore the original value of $s6 from the stack
    lw $s5, 24($sp)     # Restore the original value of $s5 from the stack
    lw $s4, 28($sp)     # Restore the original value of $s4 from the stack
    lw $s3, 32($sp)     # Restore the original value of $s3 from the stack
    lw $s2, 36($sp)     # Restore the original value of $s2 from the stack
    lw $s1, 40($sp)     # Restore the original value of $s1 from the stack
    lw $s0, 44($sp)     # Restore the original value of $s0 from the stack
    lw $ra, 48($sp)     # Restore the value of the return address ($ra) from the stack
    addiu $sp, $sp, 56  # Increment the stack pointer, taking itnto account the parameter pushed onto the stack

    jr $ra      # Jump to the return address to exit the subroutine
    nop

setpixel:
    addiu $sp, $sp, -56 # Decrement the stack pointer
    sw $ra, 48($sp)     # Save the value of the return address ($ra) to the stack
    sw $s0, 44($sp)     # Save the original value of $s0 to the stack
    sw $s1, 40($sp)     # Save the original value of $s1 to the stack
    sw $s2, 36($sp)     # Save the original value of $s2 to the stack
    sw $s3, 32($sp)     # Save the original value of $s3 to the stack
    sw $s4, 28($sp)     # Save the original value of $s4 to the stack
    sw $s5, 24($sp)     # Save the original value of $s5 to the stack
    sw $s6, 20($sp)     # Save the original value of $s6 to the stack
    sw $s7, 16($sp)     # Save the original value of $s7 to the stack
    sw $t0, 12($sp)     # Save the original value of $t0 to the stack
    sw $t1, 8($sp)      # Save the original value of $t1 to the stack
    sw $t2, 4($sp)      # Save the original value of $t2 to the stack
    sw $t3, ($sp)       # Save the original value of $t3 to the stack

    slti $t7, $a0, 0xff
    beq $t7, $zero, exit2
    slti $t7, $a1, 0xff
    beq $t7, $zero, exit2
    # Store the x co-ordinate in $a0
    # Store the y co-ordinate in $a1
    # Store the colour in $a2
    lui $s7, 0xfafa
    ori $s7, $s7, 0xffff #white
    lui $s0, 12
    ori $s0, $s0, 0x2000 #s0 <- first row

    #pointad($a0) draw at xxyy
    move $t0, $a1 #get line
    move $t2, $a0 #get column
    addi $t9, $zero, 0x140
    mult $t0, $t9
    mflo $t0 #line * 320
    add $t0, $t0, $t2 #line * 320 + column
    sll $t0, $t0, 2 # pixel * 4
    add $t1, $s0, $t0
    sw $s7, 0($t1)
exit2:
    lw $t3, ($sp)       # Restore the original value of $t3 from the stack
    lw $t2, 4($sp)      # Restore the original value of $t2 from the stack
    lw $t1, 8($sp)      # Restore the original value of $t1 from the stack
    lw $t0, 12($sp)     # Restore the original value of $t0 from the stack
    lw $s7, 16($sp)     # Restore the original value of $s7 from the stack
    lw $s6, 20($sp)     # Restore the original value of $s6 from the stack
    lw $s5, 24($sp)     # Restore the original value of $s5 from the stack
    lw $s4, 28($sp)     # Restore the original value of $s4 from the stack
    lw $s3, 32($sp)     # Restore the original value of $s3 from the stack
    lw $s2, 36($sp)     # Restore the original value of $s2 from the stack
    lw $s1, 40($sp)     # Restore the original value of $s1 from the stack
    lw $s0, 44($sp)     # Restore the original value of $s0 from the stack
    lw $ra, 48($sp)     # Restore the value of the return address ($ra) from the stack
    addiu $sp, $sp, 56  # Increment the stack pointer, taking itnto account the parameter pushed onto the stack
    jr $ra      # Jump to the return address to exit the subroutine
    nop
```

- 测试
    - 测试代码

```
main:
    lui $sp, 0x000f
  ori $sp, $zero, 0xf000
    add $a0, $zero, $zero #first pixel
    jal pointad
    addi $a0, $zero, 0xA0F0 #in the middle xx: 160, tt: 240
    jal pointad
    addi $a0, $zero, 0xffff #in the max xx: 255 yy: 25l pointad
    jal pointad

    lui $a0, 0x0000
    ori $a0, $a0, 0x7f7f
    jal drawline

    lui $a0, 0x3F00
    jal drawline
    ori $a0, $a0, 0x7F7F
    jal drawline

    lui $a0, 0xFE7F
    ori $a0, $a0, 0x7f7F
    jal drawline

    loop:
    j loop
```

    - 测试结果



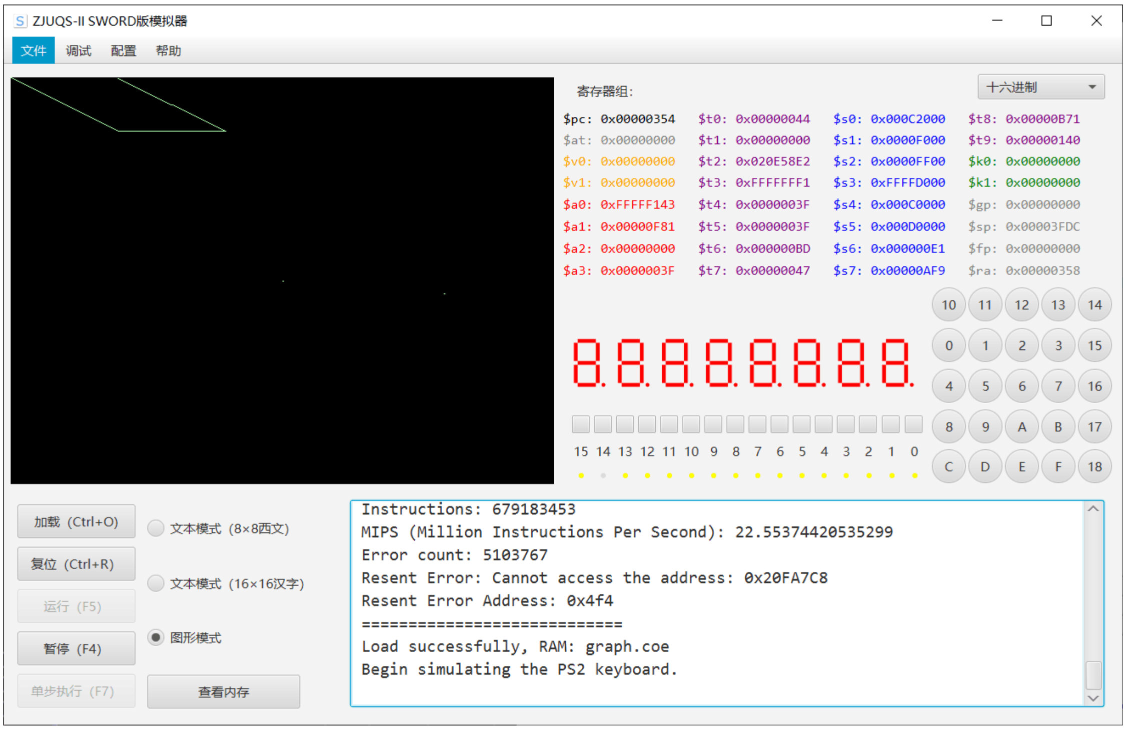
```
$pc: 0x00000064   $t0: 0x0004FEFC   $s0: 0x000C2000   $t8: 0x00000000
$at: 0x00000000   $t1: 0x00111EFC   $s1: 0xFAFA0000   $t9: 0x00000140
$v0: 0x00000000   $t2: 0x00000000   $s2: 0x00000000   $k0: 0x00000000
$v1: 0x00000000   $t3: 0x00000000   $s3: 0x00000000   $k1: 0x00000000
$a0: 0x00000080   $t4: 0x00000000   $s4: 0x00000000   $gp: 0x00000000
$a1: 0x0000007F   $t5: 0x00000000   $s5: 0x00000000   $sp: 0x0000F000
$a2: 0x00000000   $t6: 0x00000000   $s6: 0x00000000   $fp: 0x00000000
$a3: 0x0000007F   $t7: 0x00000001   $s7: 0x00000000   $ra: 0x00000060
```

```
Exit simulating the PS2 keyboard.
Begin simulating the PS2 keyboard.
Exit simulating the PS2 keyboard.
Begin simulating the PS2 keyboard.
Exit simulating the PS2 keyboard.
Begin simulating the PS2 keyboard.
Exit simulating the PS2 keyboard.
Begin simulating the PS2 keyboard.
```

- 3，指定顶点画四边形
- 实现

```
main:
    lui $sp, 0x000f
    ori $sp, $zero, 0xf000
    add $a0, $zero, $zero #first pixel

    jal pointad
    addi $a0, $zero, 0XA0F0 #in the middle xx: 160, tt: 240
    jal pointad
    addi $a0, $zero, 0xffff #in the max xx: 255 yy: 251 pointad
    jal pointad

    # lui $a0, 0x7f7f
    # ori $a0, $a0, 0xffff
    # jal drawline

    # lui $a0, 0xfe01
    # ori $a0, $a0, 0x00fe
    # jal drawline

    # lui $a0, 0xfe7f
    # ori $a0, $a0, 0x007f
    # jal drawline

    # lui $a0, 0xfe9f
    # ori $a0, $a0, 0x009f
    # jal drawline

    lui $a0,0x0000
    ori $a0,$a0,0x00f1
    lui $a1,0x3f3f
    ori $a0,$a0,0x3f7f
    jal drawbox

    # add $a0,$zero,$zero
    # ori $a0,$a0,0xf090
    # addi $a1,$zero,0x20

    j drawCircle

drawbox:
    addiu $sp, $sp, -56 # Decrement the stack pointer
    sw $ra, 48($sp)     # Save the value of the return address ($ra) to the
stack
    sw $s0, 44($sp)     # Save the original value of $s0 to the stack
    sw $s1, 40($sp)     # Save the original value of $s1 to the stack
    sw $s2, 36($sp)     # Save the original value of $s2 to the stack
    sw $s3, 32($sp)     # Save the original value of $s3 to the stack
    sw $s4, 28($sp)     # Save the original value of $s4 to the stack
    sw $s5, 24($sp)     # Save the original value of $s5 to the stack
    sw $s6, 20($sp)     # Save the original value of $s6 to the stack
    sw $s7, 16($sp)     # Save the original value of $s7 to the stack
    sw $t0, 12($sp)     # Save the original value of $t0 to the stack
    sw $t1, 8($sp)      # Save the original value of $t1 to the stack
    sw $t2, 4($sp)      # Save the original value of $t2 to the stack
    sw $t3, ($sp)       # Save the original value of $t3 to the stack

    ori $t1, $zero, 0xffff
    srl $s0, $a0, 16        # Store the x co-ordinate of point 1 in $s0
    and $s1, $s0, $t1       # Store the y co-ordinate of point 1 in $s1
    srl $s0, $a1, 16        # Store the x co-ordinate of point 2 in $s2
    and $s1, $s1, $t1       # Store the y co-ordinate of point 2 in $s3

    lui $a0, 0x3f01
    ori $a0, $a0, 0x7e3f
    jal drawline

    lui $a0, 0x0000
    ori $a0, $a0, 0x3f3f
    jal drawline

    lui $a0, 0x3f3f
    ori $a0, $a0, 0x7f3f
    jal drawline

    lw $t3, ($sp)       # Restore the original value of $t3 from the stack
    lw $t2, 4($sp)      # Restore the original value of $t2 from the stack
    lw $t1, 8($sp)      # Restore the original value of $t1 from the stack
    lw $t0, 12($sp)     # Restore the original value of $t0 from the stack
    lw $s7, 16($sp)     # Restore the original value of $s7 from the stack
    lw $s6, 20($sp)     # Restore the original value of $s6 from the stack
    lw $s5, 24($sp)     # Restore the original value of $s5 from the stack
    lw $s4, 28($sp)     # Restore the original value of $s4 from the stack
    lw $s3, 32($sp)     # Restore the original value of $s3 from the stack
    lw $s2, 36($sp)     # Restore the original value of $s2 from the stack
    lw $s1, 40($sp)     # Restore the original value of $s1 from the stack
    lw $s0, 44($sp)     # Restore the original value of $s0 from the stack
    lw $ra, 48($sp)     # Restore the value of the return address ($ra) from
the stack
    addiu $sp, $sp, 56  # Increment the stack pointer, taking itnto account
the parameter pushed onto the stack
    jr $ra      # Jump to the return address to exit the subroutine
    nop
```

- 运行结果



寄存器组:

```
$pc: 0x00000354   $t0: 0x00000044   $s0: 0x000C2000   $t8: 0x00000B71
$at: 0x00000000   $t1: 0x00000000   $s1: 0x0000F000   $t9: 0x00000140
$v0: 0x00000000   $t2: 0x020E58E2   $s2: 0x0000FF00   $k0: 0x00000000
$v1: 0x00000000   $t3: 0xFFFFFFF1   $s3: 0xFFFFD000   $k1: 0x00000000
$a0: 0xFFFFF143   $t4: 0x0000003F   $s4: 0x000C0000   $gp: 0x00000000
$a1: 0x00000F81   $t5: 0x0000003F   $s5: 0x000D0000   $sp: 0x00003FDC
$a2: 0x00000000   $t6: 0x000000BD   $s6: 0x000000E1   $fp: 0x00000000
$a3: 0x0000003F   $t7: 0x00000047   $s7: 0x00000AF9   $ra: 0x00000358
```

○ 文本模式（8×8西文）

○ 文本模式（16×16汉字）

● 图形模式

```
Instructions: 679183453
MIPS (Million Instructions Per Second): 22.55374420535299
Error count: 5103767
Resent Error: Cannot access the address: 0x20FA7C8
Resent Error Address: 0x4f4
============================
Load successfully, RAM: graph.coe
Begin simulating the PS2 keyboard.
```

- 4，指定圆心半径画圆
  - 实现

```
drawCircle:
    addi $sp, $zero, 16384
    addi $s3, $zero, 53248
    lui $s4, 12
    lui $s5,13
    lui $s1,240
    srl $s1,$s1,8
    addi $s2,$s1,0xF00
    addi $t0,$zero,255
    and $t2,$a0,$t0
    srl $a0,$a0,8
    and $t1,$a0,$t0
    srl $a0,$a0,8
    sub $t3,$t1,$a1
    add $t4,$t1,$a1
    sub $t5,$t2,$a1
    add $t6,$t2,$a1
    mult $a1,$a1
    mflo $a1
    OuterLoop:

    add $t7,$zero,$t5
    InnerLoop:
    sub $s6,$t3,$t1
    slt $t0,$s6,$zero
    beq $t0,$zero,S6_big0
    sub $s6,$zero,$s6
    S6_big0:
    sub $s7,$t7,$t2
    slt $t0,$s7,$zero
    beq $t0,$zero,S7_big0
```

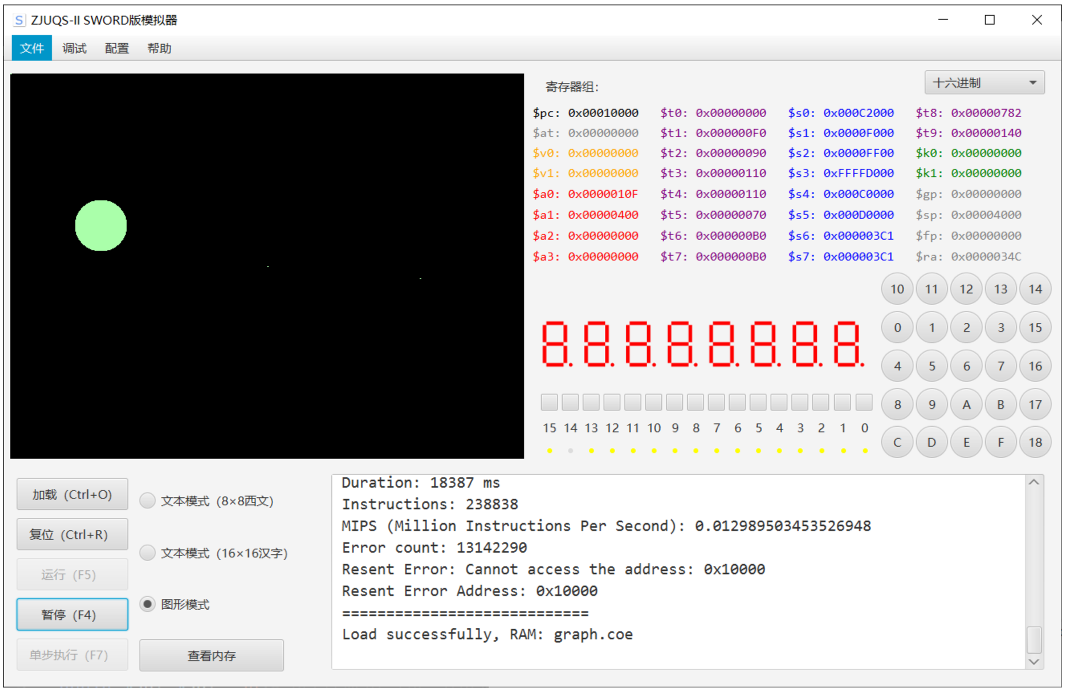
```
sub $s7,$zero,$s7
S7_big0:
mult $s6,$s6
mflo $s6
mult $s7,$s7
mflo $s7
add $t8,$s7,$s6
slt $t0,$t8,$a1
beq $t0,$zero,circle_skip
add $a0,$zero,$zero
add $a0,$zero,$t3
sll $a0,$a0,8
add $a0,$a0,$t7
jal drawCirclePoint
circle_skip:
addi $t7,$t7,1
slt $t0,$t7,$t6
bne $t0,$zero,InnerLoop

addi $t3,$t3,1
slt $t0,$t3,$t4
bne $t0,$zero,OuterLoop
j end

drawCirclePoint:
addi $sp,$sp,65500
sw $t0,($sp)
sw $t1,4($sp)
sw $t2,8($sp)
sw $t3,12($sp)
sw $t4,16($sp)
sw $t5,20($sp)
sw $t6,24($sp)
sw $t7,28($sp)
sw $t8,32($sp)
add $v0,$zero,$zero
add $t1,$zero,$zero
add $t2,$zero,$zero
addi $t0,$zero,255
and $t0,$a0,$t0 //t0 = y
srl $a0,$a0,8 //a0 = x
addi $t3,$zero,640
mult $t0,$t3
mflo $t3
add $t2,$t3,$a0
sll $t2,$t2,1
add $t2,$s5,$t2
lui $t5,0xFAFA
sw $t5,($t2)
lw $t0,($sp)
lw $t1,4($sp)
lw $t2,8($sp)
lw $t3,12($sp)
lw $t4,16($sp)
lw $t5,20($sp)
lw $t6,24($sp)
lw $t7,28($sp)
lw $t8,32($sp)
```

```
    addi $sp,$sp,36
    jr $ra
```

- 测试
  - 测试代码

```
main:
    lui $sp, 0x000f
    ori $sp, $zero, 0xf000
    add $a0, $zero, $zero #first pixel

    jal pointad
    addi $a0, $zero, 0xA0F0 #in the middle xx: 160, tt: 240
    jal pointad
    addi $a0, $zero, 0xffff #in the max xx: 255 yy: 251 pointad
    jal pointad
    add $a0,$zero,$zero
    ori $a0,$a0,0xf090
    addi $a1,$zero,0x20

    j drawCircle
```

  - 测试结果



○ 文本模式（8×8西文）
○ 文本模式（16×16汉字）
● 图形模式

```
$pc: 0x00010000   $t0: 0x00000000   $s0: 0x000C2000   $t8: 0x00000782
$at: 0x00000000   $t1: 0x000000F0   $s1: 0x0000F000   $t9: 0x00000140
$v0: 0x00000000   $t2: 0x00000090   $s2: 0x0000FF00   $k0: 0x00000000
$v1: 0x00000000   $t3: 0x00000110   $s3: 0xFFFFD000   $k1: 0x00000000
$a0: 0x0000010F   $t4: 0x00000110   $s4: 0x000C0000   $gp: 0x00000000
$a1: 0x00000400   $t5: 0x00000070   $s5: 0x000D0000   $sp: 0x00004000
$a2: 0x00000000   $t6: 0x000000B0   $s6: 0x000003C1   $fp: 0x00000000
$a3: 0x00000000   $t7: 0x000000B0   $s7: 0x000003C1   $ra: 0x0000034C
```

```
Duration: 18387 ms
Instructions: 238838
MIPS (Million Instructions Per Second): 0.012989503453526948
Error count: 13142290
Resent Error: Cannot access the address: 0x10000
Resent Error Address: 0x10000
============================
Load successfully, RAM: graph.coe
```