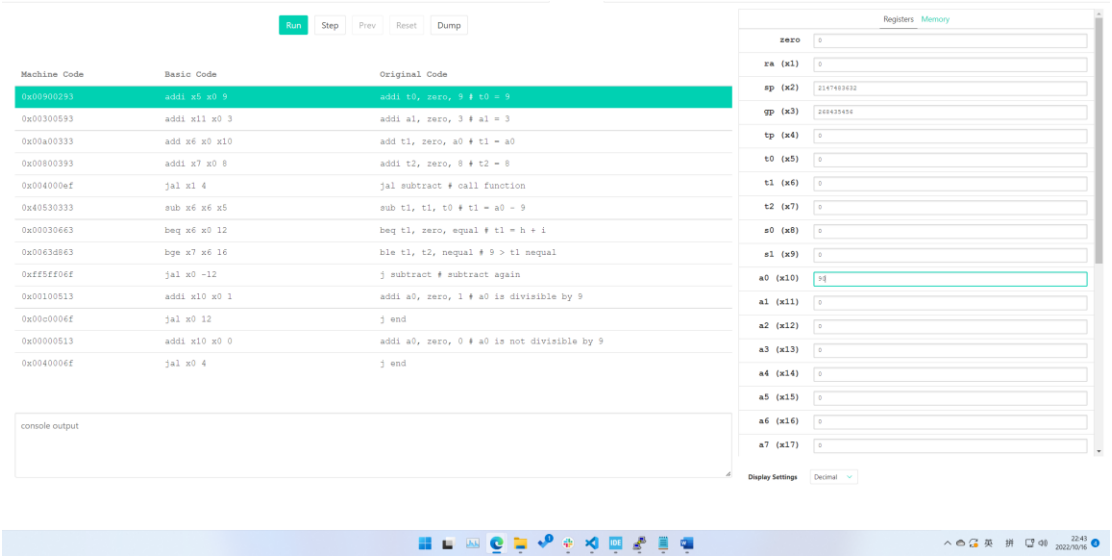
I spent 2h for this lab.

Divide by 9
Code:

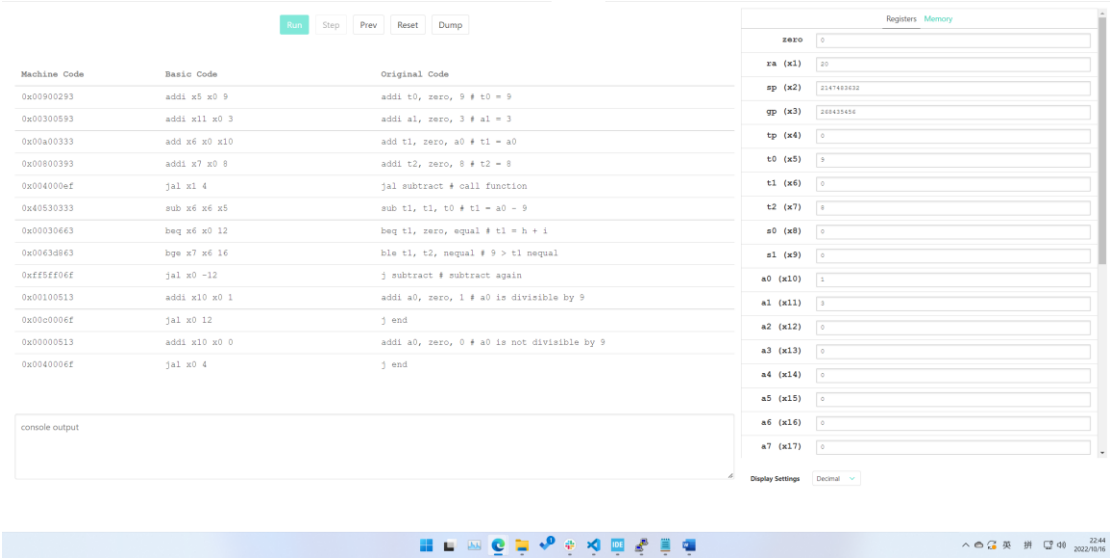
```
main:
addi t0, zero, 9  # t0 = 9
addi a1, zero, 3  # a1 = 3
add t1, zero, a0  # t1 = a0
addi t2, zero, 8  # t2 = 8
jal subtract # call function

subtract:
sub  t1, t1, t0   # t1 = a0 - 9
beq  t1, zero, equal   # t1 = h + i
ble  t1, t2, nequal # 9 > t1 nequal
j subtract # subtract again

equal:
addi a0, zero, 1 # a0 is divisible by 9
j end

nequal:
addi a0, zero, 0 # a0 is not divisible by 9
j end

end:
#Finished
```

Simulation:

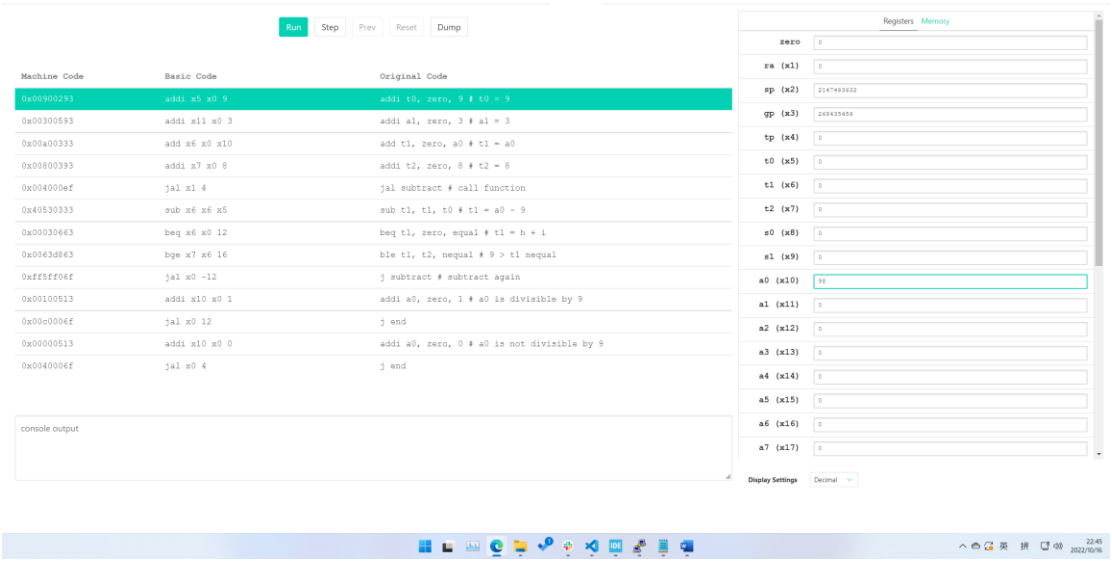
a0 = 90



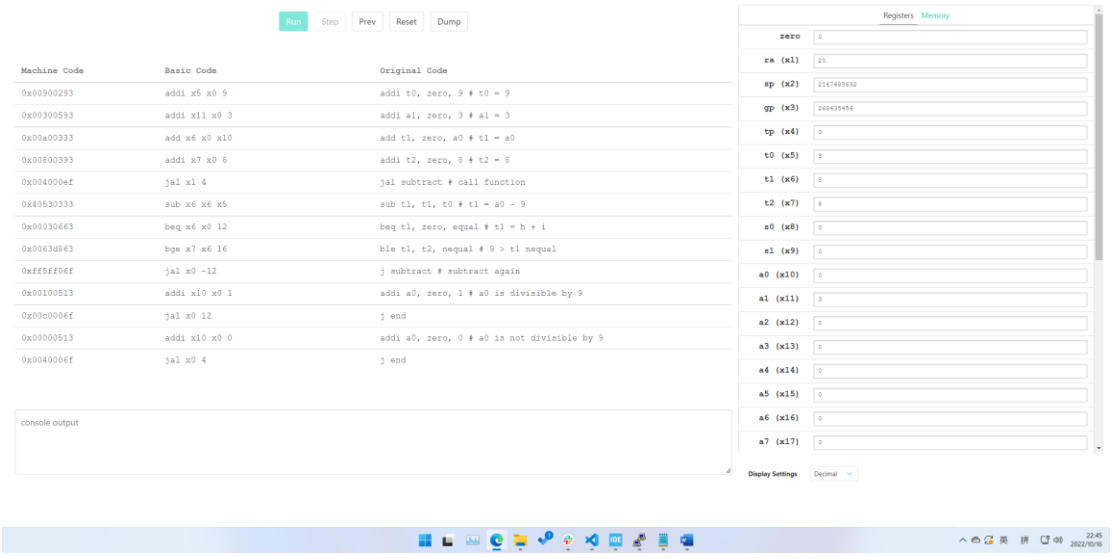
a0 = 1, divisible



a0 = 98



a0 = 0 not divisible



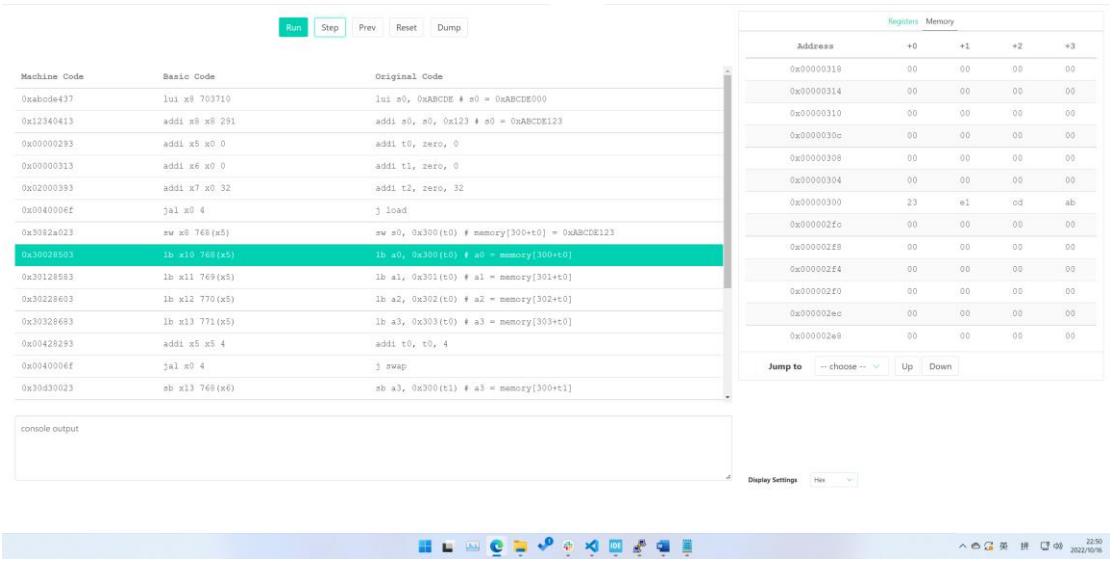
big2little

code:

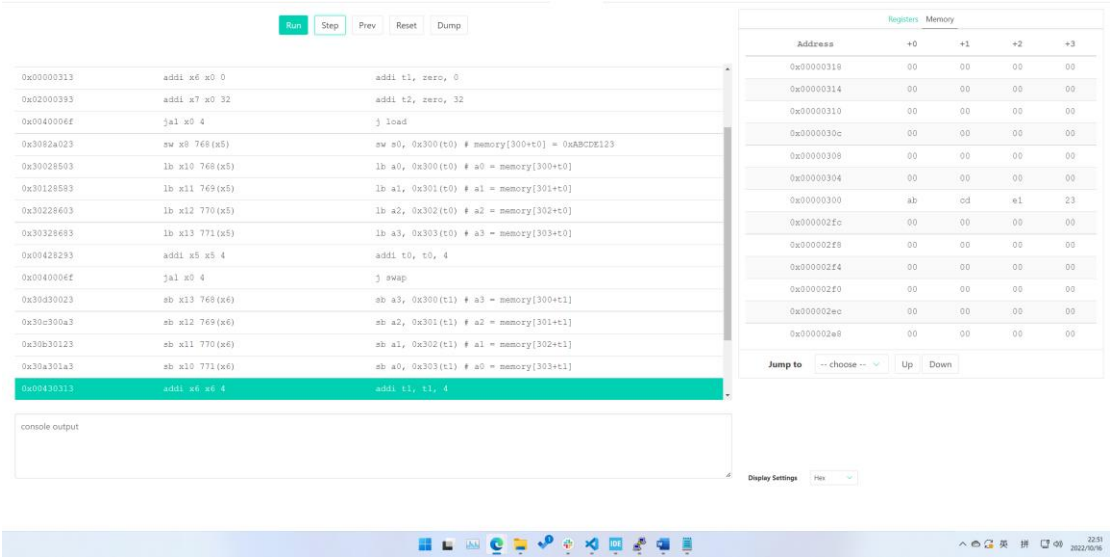
```
lui s0, 0xABCDE # s0 = 0xABCDE000
addi s0, s0, 0x123 # s0 = 0xABCDE123
addi t0, zero, 0
addi t1, zero, 0
addi t2, zero, 32


main:
j load

load:
sw s0, 0x300(t0) # memory[300+t0] = 0xABCDE123
lb a0, 0x300(t0) # a0 = memory[300+t0]
lb a1, 0x301(t0) # a1 = memory[301+t0]
lb a2, 0x302(t0) # a2 = memory[302+t0]
lb a3, 0x303(t0) # a3 = memory[303+t0]
addi t0, t0, 4
j swap

swap:
sb a3, 0x300(t1) # a3 = memory[300+t1]
sb a2, 0x301(t1) # a2 = memory[301+t1]
sb a1, 0x302(t1) # a1 = memory[302+t1]
sb a0, 0x303(t1) # a0 = memory[303+t1]
addi t1, t1, 4
j eightwords # word counter

eightwords:
beq t0, t2, finish
j load

finish:
```

Simulation:

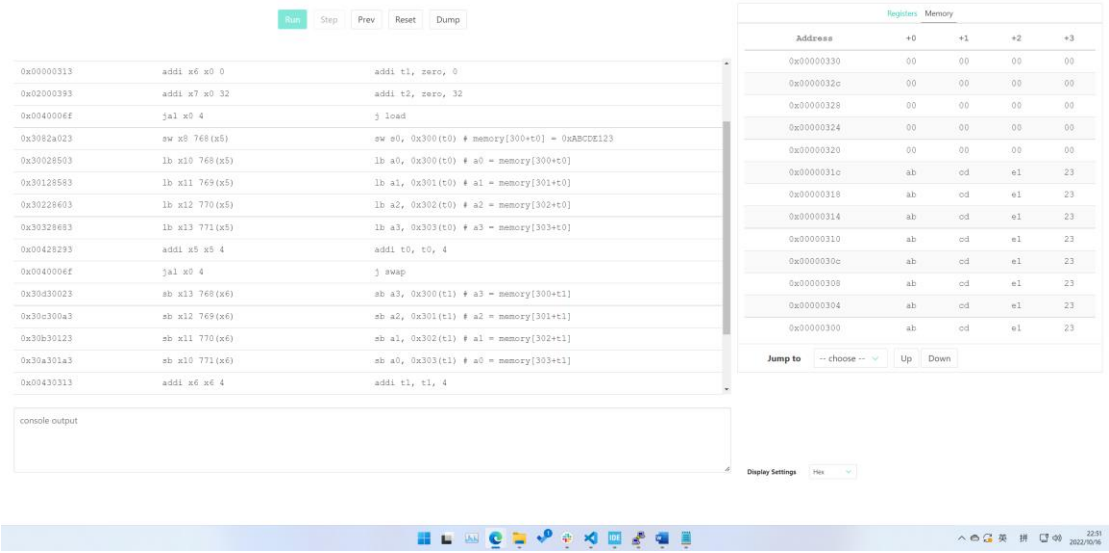
First word big-endian



First word small-endian



Finish


Run
Step
Prev
Reset
Dump
0x00000313 addi x6 x0 0 addi t1, zero, 0
0x02000393 addi x7 x0 32 addi t2, zero, 32
0x0040006f jal x0 4 j load
0x3082a023 sw x8 768(x5) sw s0, 0x300(t0) # memory[300+t0] = 0xABCDE123
0x30028503 lb x10 768(x5) lb a0, 0x300(t0) # a0 = memory[300+t0]
0x30128583 lb x11 769(x5) lb a1, 0x301(t0) # a1 = memory[301+t0]
0x30228603 lb x12 770(x5) lb a2, 0x302(t0) # a2 = memory[302+t0]
0x30328683 lb x13 771(x5) lb a3, 0x303(t0) # a3 = memory[303+t0]
0x00428293 addi x5 x5 4 addi t0, t0, 4
0x0040006f jal x0 4 j swap
0x30d30023 sb x13 768(x6) sb a3, 0x300(t1) # a3 = memory[300+t1]
0x30c300a3 sb x12 769(x6) sb a2, 0x301(t1) # a2 = memory[301+t1]
0x30b30123 sb x11 770(x6) sb a1, 0x302(t1) # a1 = memory[302+t1]
0x30a301a3 sb x10 771(x6) sb a0, 0x303(t1) # a0 = memory[303+t1]
0x00430313 addi x6 x6 4 addi t1, t1, 4
console output
Registers
Memory
Address +0 +1 +2 +3
0x00000330 00 00 00 00
0x0000032c 00 00 00 00
0x00000328 00 00 00 00
0x00000324 00 00 00 00
0x00000320 00 00 00 00
0x0000031c ab cd e1 23
0x00000318 ab cd e1 23
0x00000314 ab cd e1 23
0x00000310 ab cd e1 23
0x0000030c ab cd e1 23
0x00000308 ab cd e1 23
0x00000304 ab cd e1 23
0x00000300 ab cd e1 23
Jump to
-- choose --
Up
Down
Display Settings
Hex

Bubblesort

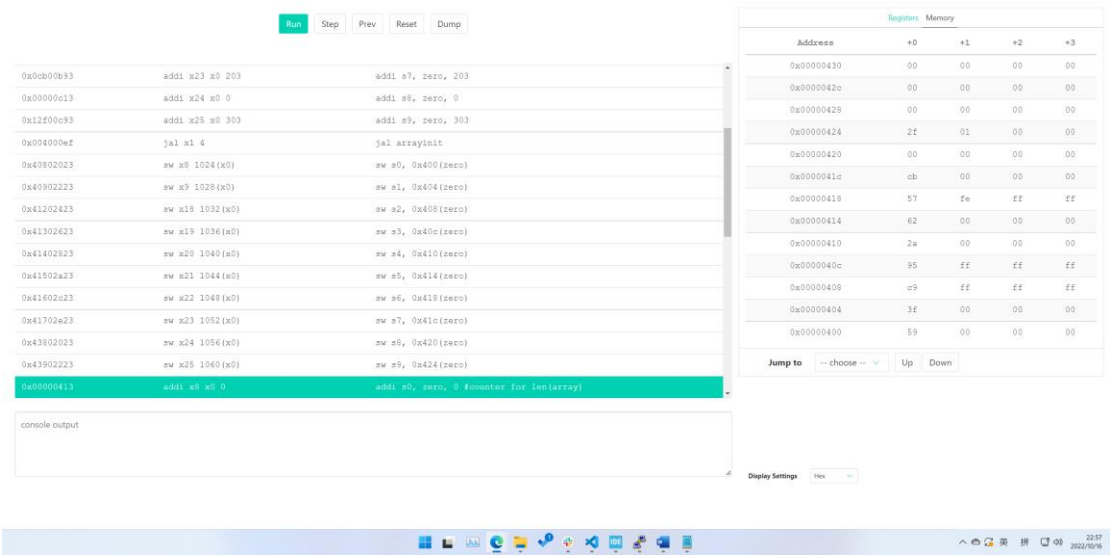
Code:

```
addi s0, zero, 89
addi s1, zero, 63
addi s2, zero, -55
addi s3, zero, -107
addi s4, zero, 42
addi s5, zero, 98
addi s6, zero, -425
addi s7, zero, 203
addi s8, zero, 0
addi s9, zero, 303

main:
jal arrayinit

arrayinit: #initialize array
sw s0, 0x400(zero)
sw s1, 0x404(zero)
sw s2, 0x408(zero)
sw s3, 0x40c(zero)
sw s4, 0x410(zero)
sw s5, 0x414(zero)
sw s6, 0x418(zero)
sw s7, 0x41c(zero)
sw s8, 0x420(zero)
sw s9, 0x424(zero)

addi s0, zero, 0 #counter for len(array)
addi s1, zero, 0 #counter for element in array
addi s2, zero, 36
addi s3, zero, 10
j sortcount

sortcount:
addi s1, zero, 0
addi s0, s0, 1
beq s0, s3, finish # if the sort has gone 9 times
j bubblesort # start bubblesort

```

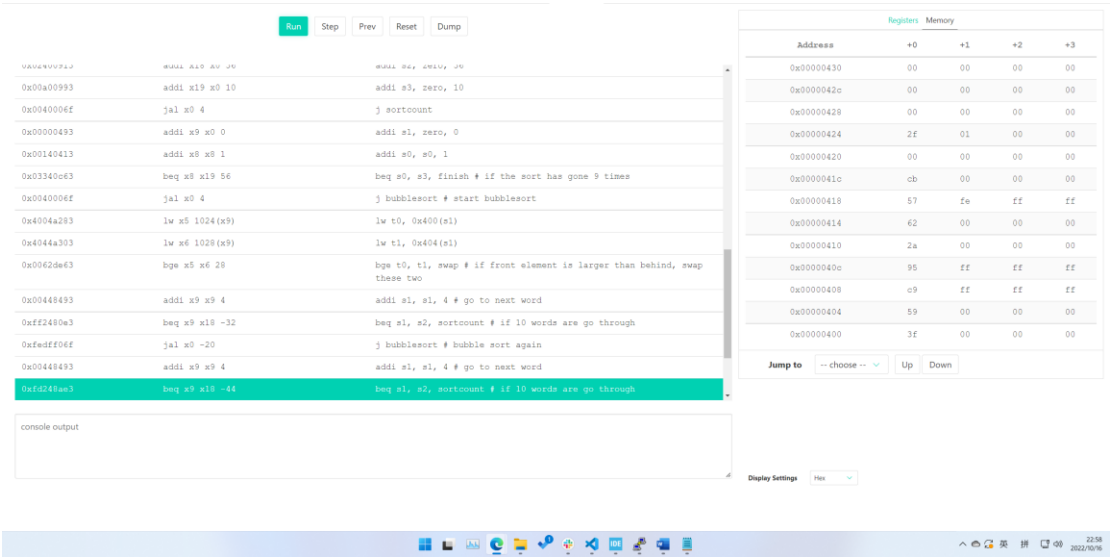
```
sw s5, 0x414(zero)
sw s6, 0x418(zero)
sw s7, 0x41c(zero)
sw s8, 0x420(zero)
sw s9, 0x424(zero)

addi s0, zero, 0 #counter for len(array)
addi s1, zero, 0 #counter for element in array
addi s2, zero, 36
addi s3, zero, 10
j sortcount

sortcount:
addi s1, zero, 0
addi s0, s0, 1
beq s0, s3, finish # if the sort has gone 9 times
j bubblesort # start bubblesort

bubblesort:
lw t0, 0x400(s1)
lw t1, 0x404(s1)
bge t0, t1, swap # if front element is larger than behind, swap these two
addi s1, s1, 4 # go to next word
beq s1, s2, sortcount # if 10 words are go through
j bubblesort # bubble sort again

afterswap:
addi s1, s1, 4 # go to next word
beq s1, s2, sortcount # if 10 words are go through
j bubblesort

swap: # swap front and behind element
sw t1, 0x400(s1)
sw t0, 0x404(s1)
j afterswap

finish:

```

Simulation:

Before bubble sort:



After first bubble sort:



Finish:

Run | Step | Prev | Reset | Dump

0x00000493	addi x9 x0 0	addi s1, zero, 0
0x00140413	addi x8 x8 1	addi s0, s0, 1
0x03340c63	beq x8 x19 56	beq s0, s3, finish # if the sort has gone 9 times
0x0040006f	jal x0 4	j bubblesort # start bubblesort
0x4004a283	lw x5 1024(x9)	lw t0, 0x400(s1)
0x4044a303	lw x6 1028(x9)	lw t1, 0x404(s1)
0x0062de63	bge x5 x6 28	bge t0, t1, swap # if front element is larger than behind, swap these two
0x00448493	addi x9 x9 4	addi s1, s1, 4 # go to next word
0xff2480e3	beq x9 x18 -32	beq s1, s2, sortcount # if 10 words are go through
0xfedff06f	jal x0 -20	j bubblesort # bubble sort again
0x00448493	addi x9 x9 4	addi s1, s1, 4 # go to next word
0xfd248ae3	beq x9 x18 -44	beq s1, s2, sortcount # if 10 words are go through
0xfe1ff06f	jal x0 -32	j bubblesort
0x4064a023	sw x6 1024(x9)	sw t1, 0x400(s1)

console output

Registers | Memory

Address	+0	+1	+2	+3
0x00000430	00	00	00	00
0x0000042c	00	00	00	00
0x00000428	00	00	00	00
0x00000424	2f	01	00	00
0x00000420	cb	00	00	00
0x0000041c	62	00	00	00
0x00000418	59	00	00	00
0x00000414	3f	00	00	00
0x00000410	2a	00	00	00
0x0000040c	00	00	00	00
0x00000408	c9	ff	ff	ff
0x00000404	95	ff	ff	ff
0x00000400	57	fe	ff	ff

Jump to -- choose -- | Up | Down

Display Settings Hex