



A Browser-based RISC-V Simulator

Adaptive and Secure Computing Systems (ASCS) Laboratory

BRISC-V Simulator

- BRISC-V Simulator lets you:
 - Run RISC-V assembly code in the browser
 - Debug hand-written assembly
 - Run code until completion or until it hits a breakpoint
 - Step through execution, instruction-by-instruction
 - View the state of memory and registers at each step
 - View how each instruction is constructed – opcodes, registers, immediate values etc.

<https://ascslab.org/research/briscv/simulator/simulator.html>

Let's get Familiar with the GUI

BRISC-V Home

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

C Source Code Pane

Not interesting for this class

Let's you compile C to RISC-V assembly

Console

```
***** Parser Output *****
Parsing successful!
```

Console Pane

Shows messages from the compiler and the simulator
Also used for system calls – let's you input and print values

BRISC-V Simulator

RISC-V Assembly

```
0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
gcd:
18     addi sp,sp,-48
19     sw ra,44(sp)
20     sw s0,40(sp)
21     addi s0,sp,48
22     sw a0,-36($0)
23     sw a1,-40($0)
24     lw a5,-36($0)
11     lw a4,-40($0)
12     bne a4,a5,.L2
13     lw a4,-36($0)
14     lw a5,-40($0)
15     j .L3
.L2:
16     lw a4,-36($0)
17     lw a5,-40($0)
```

RISC-V Assembly Pane

Let's you run assembly step-by-step

ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

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

Register & Memory Pane

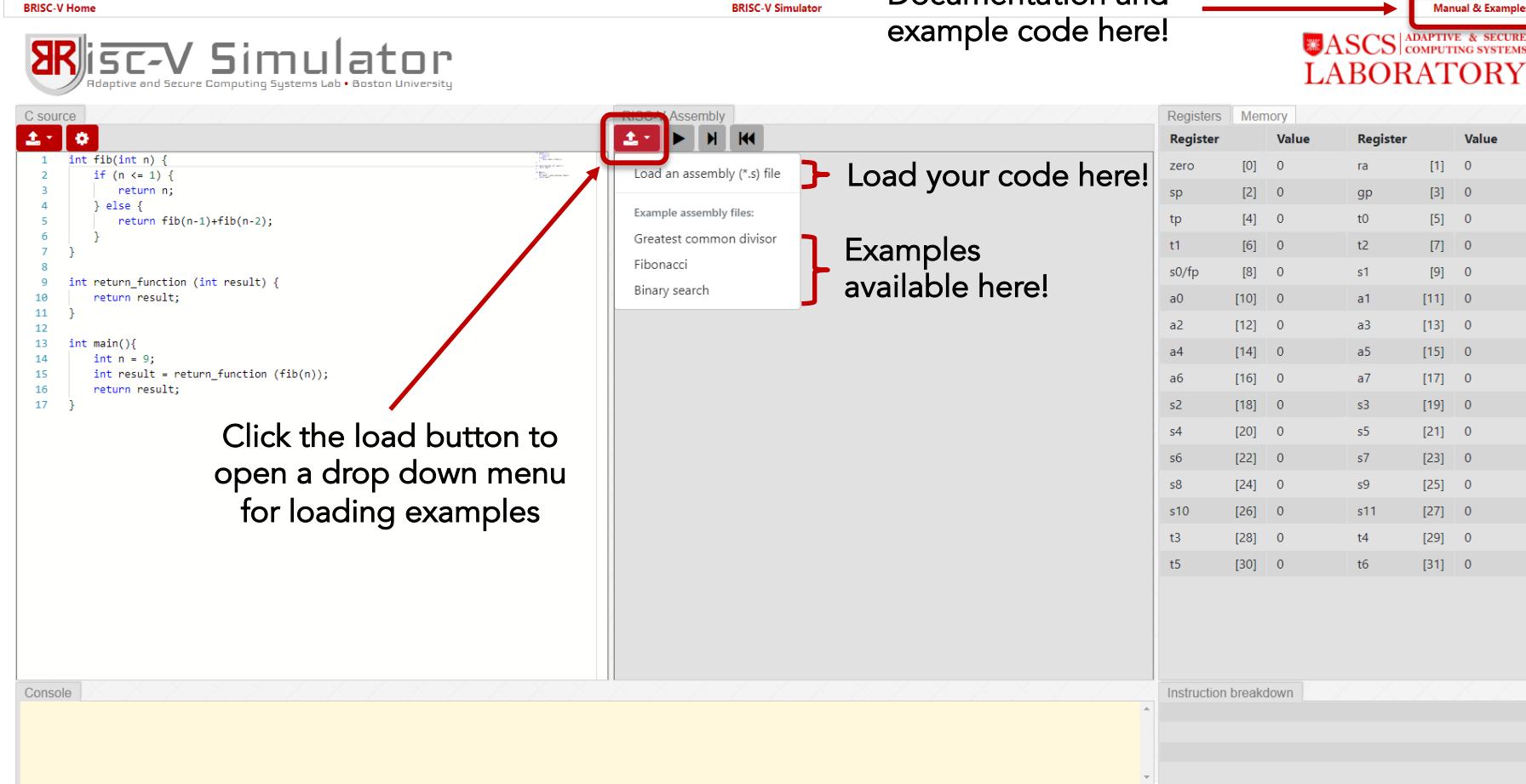
Shows the state of registers and memory

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
000000000000	00000	000	00000	00000	0010011				

Instruction Breakdown Pane

Don't have valid RISC-V assembly code to start with?



The screenshot shows the BRISC-V Simulator interface. On the left, the C source code for a Fibonacci function is displayed:

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

A red arrow points from the text "Click the load button to open a drop down menu for loading examples" to the "Load an assembly (*.s) file" button in the RISC-V Assembly panel, which is highlighted with a red box.

The RISC-V Assembly panel shows the following:

- Load an assembly (*.s) file
- Example assembly files:
 - Greatest common divisor
 - Fibonacci
 - Binary search

Documentation and example code here! → [Manual & Examples](#)

ASCS | ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Registers	Memory		
Register	Value	Register	Value
zero	[0]	ra	[1]
sp	[2]	gp	[3]
tp	[4]	t0	[5]
t1	[6]	t2	[7]
s0/fp	[8]	s1	[9]
a0	[10]	a1	[11]
a2	[12]	a3	[13]
a4	[14]	a5	[15]
a6	[16]	a7	[17]
s2	[18]	s3	[19]
s4	[20]	s5	[21]
s6	[22]	s7	[23]
s8	[24]	s9	[25]
s10	[26]	s11	[27]
t3	[28]	t4	[29]
t5	[30]	t6	[31]

Don't have valid RISC-V assembly code to start with?



BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

RISC-V Assembly

Load an assembly (*.s) file

Example assembly files:
Greatest common divisor
Fibonacci
Binary search

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

Registers Memory

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

Console

Instruction breakdown

ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Load the GCD example

Kernel and User Instructions

BRISC-V Home
BRISC-V Simulator
Manual & Examples

BRISC-V Simulator

Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

RISC-V Assembly

```
0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
gcd:
18 addi sp,sp,-48
19 sw ra,44(sp)
20 sw s0,40(sp)
21 addi s0,sp,48
22 sw a0,-36($0)
23 sw a1,-40($0)
24 lw a4,-36($0)
25 lw a5,-40($0)
26 lw a5,-40($0)
27 bne a4,a5,.L2
28 lw a5,-36($0)
29 lw a5,-20($0)
30 j .L3
.L2:
31 lw a4,-36($0)
32 lw a5,-40($0)
33 b10 a4,-56 .L4
```

Our code got loaded!

The console says that it parsed the file without problems
If there were problems, they would pop up here

***** Parser Output *****
Parsing successful!

BRISC-V Simulator

ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Registers	Memory		
Register	Value	Register	Value
zero	[0]	ra	[1]
sp	[2]	gp	[3]
tp	[4]	t0	[5]
t1	[6]	t2	[7]
s0/fp	[8]	s1	[9]
a0	[10]	a1	[11]
a2	[12]	a3	[13]
a4	[14]	a5	[15]
a6	[16]	a7	[17]
s2	[18]	s3	[19]
s4	[20]	s5	[21]
s6	[22]	s7	[23]
s8	[24]	s9	[25]
s10	[26]	s11	[27]
t3	[28]	t4	[29]
t5	[30]	t6	[31]

Instruction breakdown									
31	20	19	15	14	12	11	7	6	0
imm		rs1		funct3	rd			opcode	
000000000000	00000	000		00000		0010011			

Kernel and User Instructions

BRISC-V Home

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

RISC-V Assembly

```
1 addi zero,zero,0
2 kernel:
3     addi sp,zero,1536
4     call main
5     addi zero,zero,0
6     mv s1,a0
7     addi zero,zero,0
8     addi zero,zero,0
9     auipc ra,0x0
10    jalr ra,0(ra)
11    addi zero,zero,0
12    addi zero,zero,0
13
14    .file "gcd.c"
15    .option nopic
16    .text
17    .align 2
18    .globl gcd
19    .type gcd, @function
20
21    gcd:
22        addi sp,sp,-48
23        sw ra,44(sp)
24        sw s0,40(sp)
25        addi s0,sp,48
26        sw a0,-36($0)
27        sw a1,-40($0)
28        lw a4,-36($0)
29        lw a5,-40($0)
30        bne a4,a5,.L2
31        lw a5,-36($0)
32        sw a5,-20($0)
33        j .L3
34
35    .L2:
36        lw a4,-36($0)
37        lw a5,-40($0)
38        bne a4,a5,.L4
```

Console

```
***** Parser Output *****
Parsing successful!
```

BRISC-V Simulator

Manual & Examples

ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Registers	Memory		
Register	Value	Register	Value
zero	[0] 0	ra	[1] 0
sp	[2] 0	gp	[3] 0
tp	[4] 0	t0	[5] 0
t1	[6] 0	t2	[7] 0
s0/fp	[8] 0	s1	[9] 0
a0	[10] 0	a1	[11] 0
a2	[12] 0	a3	[13] 0
a4	[14] 0	a5	[15] 0
a6	[16] 0	a7	[17] 0
s2	[18] 0	s3	[19] 0
s4	[20] 0	s5	[21] 0
s6	[22] 0	s7	[23] 0
s8	[24] 0	s9	[25] 0
s10	[26] 0	s11	[27] 0
t3	[28] 0	t4	[29] 0
t5	[30] 0	t6	[31] 0

Instruction breakdown									
31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					
000000000000	00000	000	00000	0010011					

Grey instructions
are kernel
instructions

They setup some registers like
the stack pointer, and jump to
label "main"

Kernel and User Instructions

BRISC-V Home

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```

1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

White instructions are user instructions

All the assembly you write will be here

BRISC-V Simulator

RISC-V Assembly

```

0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
18 gcd:
19     addi sp,sp,-48
20     sw ra,44(sp)
21     sw s0,40(sp)
22     addi s0,sp,48
23     sw a0,-36($0)
24     sw a1,-40($0)
25     lw a4,-36($0)
26     lw a5,-40($0)
27     bne a4,a5,.L2
28     lw a5,-36($0)
29     sw a5,-20($0)
30     j .L3
31 .L2:
32     lw a4,-36($0)
33     lw a5,-40($0)
34     bne a4,a5,.L4
35 .L3:
36     lw a4,-36($0)
37     lw a5,-40($0)
38     bne a4,a5,.L4
39 .L4:
40     lw a4,-36($0)
41     lw a5,-40($0)
42     bne a4,a5,.L4
43 .L5:
44     lw a4,-36($0)
45     lw a5,-40($0)
46     bne a4,a5,.L4
47 .L6:
48     lw a4,-36($0)
49     lw a5,-40($0)
50     bne a4,a5,.L4
51 .L7:
52     lw a4,-36($0)
53     lw a5,-40($0)
54     bne a4,a5,.L4
55 .L8:
56     lw a4,-36($0)
57     lw a5,-40($0)
58     bne a4,a5,.L4
59 .L9:
60     lw a4,-36($0)
61     lw a5,-40($0)
62     bne a4,a5,.L4
63 .L10:
64     lw a4,-36($0)
65     lw a5,-40($0)
66     bne a4,a5,.L4
67 .L11:
68     lw a4,-36($0)
69     lw a5,-40($0)
70     bne a4,a5,.L4
71 .L12:
72     lw a4,-36($0)
73     lw a5,-40($0)
74     bne a4,a5,.L4
75 .L13:
76     lw a4,-36($0)
77     lw a5,-40($0)
78     bne a4,a5,.L4
79 .L14:
80     lw a4,-36($0)
81     lw a5,-40($0)
82     bne a4,a5,.L4
83 .L15:
84     lw a4,-36($0)
85     lw a5,-40($0)
86     bne a4,a5,.L4
87 .L16:
88     lw a4,-36($0)
89     lw a5,-40($0)
90     bne a4,a5,.L4
91 .L17:
92     lw a4,-36($0)
93     lw a5,-40($0)
94     bne a4,a5,.L4
95 .L18:
96     lw a4,-36($0)
97     lw a5,-40($0)
98     bne a4,a5,.L4
99 .L19:
100    lw a4,-36($0)
101    lw a5,-40($0)
102    bne a4,a5,.L4
103 .L20:
104    lw a4,-36($0)
105    lw a5,-40($0)
106    bne a4,a5,.L4
107 .L21:
108    lw a4,-36($0)
109    lw a5,-40($0)
110    bne a4,a5,.L4
111 .L22:
112    lw a4,-36($0)
113    lw a5,-40($0)
114    bne a4,a5,.L4
115 .L23:
116    lw a4,-36($0)
117    lw a5,-40($0)
118    bne a4,a5,.L4
119 .L24:
120    lw a4,-36($0)
121    lw a5,-40($0)
122    bne a4,a5,.L4
123 .L25:
124    lw a4,-36($0)
125    lw a5,-40($0)
126    bne a4,a5,.L4
127 .L26:
128    lw a4,-36($0)
129    lw a5,-40($0)
130    bne a4,a5,.L4
131 .L27:
132    lw a4,-36($0)
133    lw a5,-40($0)
134    bne a4,a5,.L4
135 .L28:
136    lw a4,-36($0)
137    lw a5,-40($0)
138    bne a4,a5,.L4
139 .L29:
140    lw a4,-36($0)
141    lw a5,-40($0)
142    bne a4,a5,.L4
143 .L30:
144    lw a4,-36($0)
145    lw a5,-40($0)
146    bne a4,a5,.L4
147 .L31:
148    lw a4,-36($0)
149    lw a5,-40($0)
150    bne a4,a5,.L4
151 .L32:
152    lw a4,-36($0)
153    lw a5,-40($0)
154    bne a4,a5,.L4
155 .L33:
156    lw a4,-36($0)
157    lw a5,-40($0)
158    bne a4,a5,.L4
159 .L34:
160    lw a4,-36($0)
161    lw a5,-40($0)
162    bne a4,a5,.L4
163 .L35:
164    lw a4,-36($0)
165    lw a5,-40($0)
166    bne a4,a5,.L4
167 .L36:
168    lw a4,-36($0)
169    lw a5,-40($0)
170    bne a4,a5,.L4
171 .L37:
172    lw a4,-36($0)
173    lw a5,-40($0)
174    bne a4,a5,.L4
175 .L38:
176    lw a4,-36($0)
177    lw a5,-40($0)
178    bne a4,a5,.L4
179 .L39:
180    lw a4,-36($0)
181    lw a5,-40($0)
182    bne a4,a5,.L4
183 .L40:
184    lw a4,-36($0)
185    lw a5,-40($0)
186    bne a4,a5,.L4
187 .L41:
188    lw a4,-36($0)
189    lw a5,-40($0)
190    bne a4,a5,.L4
191 .L42:
192    lw a4,-36($0)
193    lw a5,-40($0)
194    bne a4,a5,.L4
195 .L43:
196    lw a4,-36($0)
197    lw a5,-40($0)
198    bne a4,a5,.L4
199 .L44:
200    lw a4,-36($0)
201    lw a5,-40($0)
202    bne a4,a5,.L4
203 .L45:
204    lw a4,-36($0)
205    lw a5,-40($0)
206    bne a4,a5,.L4
207 .L46:
208    lw a4,-36($0)
209    lw a5,-40($0)
210    bne a4,a5,.L4
211 .L47:
212    lw a4,-36($0)
213    lw a5,-40($0)
214    bne a4,a5,.L4
215 .L48:
216    lw a4,-36($0)
217    lw a5,-40($0)
218    bne a4,a5,.L4
219 .L49:
220    lw a4,-36($0)
221    lw a5,-40($0)
222    bne a4,a5,.L4
223 .L50:
224    lw a4,-36($0)
225    lw a5,-40($0)
226    bne a4,a5,.L4
227 .L51:
228    lw a4,-36($0)
229    lw a5,-40($0)
230    bne a4,a5,.L4
231 .L52:
232    lw a4,-36($0)
233    lw a5,-40($0)
234    bne a4,a5,.L4
235 .L53:
236    lw a4,-36($0)
237    lw a5,-40($0)
238    bne a4,a5,.L4
239 .L54:
240    lw a4,-36($0)
241    lw a5,-40($0)
242    bne a4,a5,.L4
243 .L55:
244    lw a4,-36($0)
245    lw a5,-40($0)
246    bne a4,a5,.L4
247 .L56:
248    lw a4,-36($0)
249    lw a5,-40($0)
250    bne a4,a5,.L4
251 .L57:
252    lw a4,-36($0)
253    lw a5,-40($0)
254    bne a4,a5,.L4
255 .L58:
256    lw a4,-36($0)
257    lw a5,-40($0)
258    bne a4,a5,.L4
259 .L59:
260    lw a4,-36($0)
261    lw a5,-40($0)
262    bne a4,a5,.L4
263 .L60:
264    lw a4,-36($0)
265    lw a5,-40($0)
266    bne a4,a5,.L4
267 .L61:
268    lw a4,-36($0)
269    lw a5,-40($0)
270    bne a4,a5,.L4
271 .L62:
272    lw a4,-36($0)
273    lw a5,-40($0)
274    bne a4,a5,.L4
275 .L63:
276    lw a4,-36($0)
277    lw a5,-40($0)
278    bne a4,a5,.L4
279 .L64:
280    lw a4,-36($0)
281    lw a5,-40($0)
282    bne a4,a5,.L4
283 .L65:
284    lw a4,-36($0)
285    lw a5,-40($0)
286    bne a4,a5,.L4
287 .L66:
288    lw a4,-36($0)
289    lw a5,-40($0)
290    bne a4,a5,.L4
291 .L67:
292    lw a4,-36($0)
293    lw a5,-40($0)
294    bne a4,a5,.L4
295 .L68:
296    lw a4,-36($0)
297    lw a5,-40($0)
298    bne a4,a5,.L4
299 .L69:
300    lw a4,-36($0)
301    lw a5,-40($0)
302    bne a4,a5,.L4
303 .L70:
304    lw a4,-36($0)
305    lw a5,-40($0)
306    bne a4,a5,.L4
307 .L71:
308    lw a4,-36($0)
309    lw a5,-40($0)
310    bne a4,a5,.L4
311 .L72:
312    lw a4,-36($0)
313    lw a5,-40($0)
314    bne a4,a5,.L4
315 .L73:
316    lw a4,-36($0)
317    lw a5,-40($0)
318    bne a4,a5,.L4
319 .L74:
320    lw a4,-36($0)
321    lw a5,-40($0)
322    bne a4,a5,.L4
323 .L75:
324    lw a4,-36($0)
325    lw a5,-40($0)
326    bne a4,a5,.L4
327 .L76:
328    lw a4,-36($0)
329    lw a5,-40($0)
330    bne a4,a5,.L4
331 .L77:
332    lw a4,-36($0)
333    lw a5,-40($0)
334    bne a4,a5,.L4
335 .L78:
336    lw a4,-36($0)
337    lw a5,-40($0)
338    bne a4,a5,.L4
339 .L79:
340    lw a4,-36($0)
341    lw a5,-40($0)
342    bne a4,a5,.L4
343 .L80:
344    lw a4,-36($0)
345    lw a5,-40($0)
346    bne a4,a5,.L4
347 .L81:
348    lw a4,-36($0)
349    lw a5,-40($0)
350    bne a4,a5,.L4
351 .L82:
352    lw a4,-36($0)
353    lw a5,-40($0)
354    bne a4,a5,.L4
355 .L83:
356    lw a4,-36($0)
357    lw a5,-40($0)
358    bne a4,a5,.L4
359 .L84:
360    lw a4,-36($0)
361    lw a5,-40($0)
362    bne a4,a5,.L4
363 .L85:
364    lw a4,-36($0)
365    lw a5,-40($0)
366    bne a4,a5,.L4
367 .L86:
368    lw a4,-36($0)
369    lw a5,-40($0)
370    bne a4,a5,.L4
371 .L87:
372    lw a4,-36($0)
373    lw a5,-40($0)
374    bne a4,a5,.L4
375 .L88:
376    lw a4,-36($0)
377    lw a5,-40($0)
378    bne a4,a5,.L4
379 .L89:
380    lw a4,-36($0)
381    lw a5,-40($0)
382    bne a4,a5,.L4
383 .L90:
384    lw a4,-36($0)
385    lw a5,-40($0)
386    bne a4,a5,.L4
387 .L91:
388    lw a4,-36($0)
389    lw a5,-40($0)
390    bne a4,a5,.L4
391 .L92:
392    lw a4,-36($0)
393    lw a5,-40($0)
394    bne a4,a5,.L4
395 .L93:
396    lw a4,-36($0)
397    lw a5,-40($0)
398    bne a4,a5,.L4
399 .L94:
400    lw a4,-36($0)
401    lw a5,-40($0)
402    bne a4,a5,.L4
403 .L95:
404    lw a4,-36($0)
405    lw a5,-40($0)
406    bne a4,a5,.L4
407 .L96:
408    lw a4,-36($0)
409    lw a5,-40($0)
410    bne a4,a5,.L4
411 .L97:
412    lw a4,-36($0)
413    lw a5,-40($0)
414    bne a4,a5,.L4
415 .L98:
416    lw a4,-36($0)
417    lw a5,-40($0)
418    bne a4,a5,.L4
419 .L99:
420    lw a4,-36($0)
421    lw a5,-40($0)
422    bne a4,a5,.L4
423 .L100:
424    lw a4,-36($0)
425    lw a5,-40($0)
426    bne a4,a5,.L4
427 .L101:
428    lw a4,-36($0)
429    lw a5,-40($0)
430    bne a4,a5,.L4
431 .L102:
432    lw a4,-36($0)
433    lw a5,-40($0)
434    bne a4,a5,.L4
435 .L103:
436    lw a4,-36($0)
437    lw a5,-40($0)
438    bne a4,a5,.L4
439 .L104:
440    lw a4,-36($0)
441    lw a5,-40($0)
442    bne a4,a5,.L4
443 .L105:
444    lw a4,-36($0)
445    lw a5,-40($0)
446    bne a4,a5,.L4
447 .L106:
448    lw a4,-36($0)
449    lw a5,-40($0)
450    bne a4,a5,.L4
451 .L107:
452    lw a4,-36($0)
453    lw a5,-40($0)
454    bne a4,a5,.L4
455 .L108:
456    lw a4,-36($0)
457    lw a5,-40($0)
458    bne a4,a5,.L4
459 .L109:
460    lw a4,-36($0)
461    lw a5,-40($0)
462    bne a4,a5,.L4
463 .L110:
464    lw a4,-36($0)
465    lw a5,-40($0)
466    bne a4,a5,.L4
467 .L111:
468    lw a4,-36($0)
469    lw a5,-40($0)
470    bne a4,a5,.L4
471 .L112:
472    lw a4,-36($0)
473    lw a5,-40($0)
474    bne a4,a5,.L4
475 .L113:
476    lw a4,-36($0)
477    lw a5,-40($0)
478    bne a4,a5,.L4
479 .L114:
480    lw a4,-36($0)
481    lw a5,-40($0)
482    bne a4,a5,.L4
483 .L115:
484    lw a4,-36($0)
485    lw a5,-40($0)
486    bne a4,a5,.L4
487 .L116:
488    lw a4,-36($0)
489    lw a5,-40($0)
490    bne a4,a5,.L4
491 .L117:
492    lw a4,-36($0)
493    lw a5,-40($0)
494    bne a4,a5,.L4
495 .L118:
496    lw a4,-36($0)
497    lw a5,-40($0)
498    bne a4,a5,.L4
499 .L119:
500    lw a4,-36($0)
501    lw a5,-40($0)
502    bne a4,a5,.L4
503 .L120:
504    lw a4,-36($0)
505    lw a5,-40($0)
506    bne a4,a5,.L4
507 .L121:
508    lw a4,-36($0)
509    lw a5,-40($0)
510    bne a4,a5,.L4
511 .L122:
512    lw a4,-36($0)
513    lw a5,-40($0)
514    bne a4,a5,.L4
515 .L123:
516    lw a4,-36($0)
517    lw a5,-40($0)
518    bne a4,a5,.L4
519 .L124:
520    lw a4,-36($0)
521    lw a5,-40($0)
522    bne a4,a5,.L4
523 .L125:
524    lw a4,-36($0)
525    lw a5,-40($0)
526    bne a4,a5,.L4
527 .L126:
528    lw a4,-36($0)
529    lw a5,-40($0)
530    bne a4,a5,.L4
531 .L127:
532    lw a4,-36($0)
533    lw a5,-40($0)
534    bne a4,a5,.L4
535 .L128:
536    lw a4,-36($0)
537    lw a5,-40($0)
538    bne a4,a5,.L4
539 .L129:
540    lw a4,-36($0)
541    lw a5,-40($0)
542    bne a4,a5,.L4
543 .L130:
544    lw a4,-36($0)
545    lw a5,-40($0)
546    bne a4,a5,.L4
547 .L131:
548    lw a4,-36($0)
549    lw a5,-40($0)
550    bne a4,a5,.L4
551 .L132:
552    lw a4,-36($0)
553    lw a5,-40($0)
554    bne a4,a5,.L4
555 .L133:
556    lw a4,-36($0)
557    lw a5,-40($0)
558    bne a4,a5,.L4
559 .L134:
560    lw a4,-36($0)
561    lw a5,-40($0)
562    bne a4,a5,.L4
563 .L135:
564    lw a4,-36($0)
565    lw a5,-40($0)
566    bne a4,a5,.L4
567 .L136:
568    lw a4,-36($0)
569    lw a5,-40($0)
570    bne a4,a5,.L4
571 .L137:
572    lw a4,-36($0)
573    lw a5,-40($0)
574    bne a4,a5,.L4
575 .L138:
576    lw a4,-36($0)
577    lw a5,-40($0)
578    bne a4,a5,.L4
579 .L139:
580    lw a4,-36($0)
581    lw a5,-40($0)
582    bne a4,a5,.L4
583 .L140:
584    lw a4,-36($0)
585    lw a5,-40($0)
586    bne a4,a5,.L4
587 .L141:
588    lw a4,-36($0)
589    lw a5,-40($0)
590    bne a4,a5,.L4
591 .L142:
592    lw a4,-36($0)
593    lw a5,-40($0)
594    bne a4,a5,.L4
595 .L143:
596    lw a4,-36($0)
597    lw a5,-40($0)
598    bne a4,a5,.L4
599 .L144:
600    lw a4,-36($0)
601    lw a5,-40($0)
602    bne a4,a5,.L4
603 .L145:
604    lw a4,-36($0)
605    lw a5,-40($0)
606    bne a4,a5,.L4
607 .L146:
608    lw a4,-36($0)
609    lw a5,-40($0)
610    bne a4,a5,.L4
611 .L147:
612    lw a4,-36($0)
613    lw a5,-40($0)
614    bne a4,a5,.L4
615 .L148:
616    lw a4,-36($0)
617    lw a5,-40($0)
618    bne a4,a5,.L4
619 .L149:
620    lw a4,-36($0)
621    lw a5,-40($0)
622    bne a4,a5,.L4
623 .L150:
624    lw a4,-36($0)
625    lw a5,-40($0)
626    bne a4,a5,.L4
627 .L151:
628    lw a4,-36($0)
629    lw a5,-40($0)
630    bne a4,a5,.L4
631 .L152:
632    lw a4,-36($0)
633    lw a5,-40($0)
634    bne a4,a5,.L4
635 .L153:
636    lw a4,-36($0)
637    lw a5,-40($0)
638    bne a4,a5,.L4
639 .L154:
640    lw a4,-36($0)
641    lw a5,-40($0)
642    bne a4,a5,.L4
643 .L155:
644    lw a4,-36($0)
645    lw a5,-40($0)
646    bne a4,a5,.L4
647 .L156:
648    lw a4,-36($0)
649    lw a5,-40($0)
650    bne a4,a5,.L4
651 .L157:
652    lw a4,-36($0)
653    lw a5,-40($0)
654    bne a4,a5,.L4
655 .L158:
656    lw a4,-36($0)
657    lw a5,-40($0)
658    bne a4,a5,.L4
659 .L159:
660    lw a4,-36($0)
661    lw a5,-40($0)
662    bne a4,a5,.L4
663 .L160:
664    lw a4,-36($0)
665    lw a5,-40($0)
666    bne a4,a5,.L4
667 .L161:
668    lw a4,-36($0)
669    lw a5,-40($0)
670    bne a4,a5,.L4
671 .L162:
672    lw a4,-36($0)
673    lw a5,-40($0)
674    bne a4,a5,.L4
675 .L163:
676    lw a4,-36($0)
677    lw a5,-40($0)
678    bne a4,a5,.L4
679 .L164:
680    lw a4,-36($0)
681    lw a5,-40($0)
682    bne a4,a5,.L4
683 .L165:
684    lw a4,-36($0)
685    lw a5,-40($0)
686    bne a4,a5,.L4
687 .L166:
688    lw a4,-36($0)
689    lw a5,-40($0)
690    bne a4,a5,.L4
691 .L167:
692    lw a4,-36($0)
693    lw a5,-40($0)
694    bne a4,a5,.L4
695 .L168:
696    lw a4,-36($0)
697    lw a5,-40($0)
698    bne a4,a5,.L4
699 .L169:
700    lw a4,-36($0)
701    lw a5,-40($0)
702    bne a4,a5,.L4
703 .L170:
704    lw a4,-36($0)
705    lw a5,-40($0)
706    bne a4,a5,.L4
707 .L171:
708    lw a4,-36($0)
709    lw a5,-40($0)
710    bne a4,a5,.L4
711 .L172:
712    lw a4,-36($0)
713    lw a5,-40($0)
714    bne a4,a5,.L4
715 .L173:
716    lw a4,-36($0)
717    lw a5,-40($0)
718    bne a4,a5,.L4
719 .L174:
720    lw a4,-36($0)
721    lw a5,-40($0)
722    bne a4,a5,.L4
723 .L175:
724    lw a4,-36($0)
725    lw a5,-40($0)
726    bne a4,a5,.L4
727 .L176:
728    lw a4,-36($0)
729    lw a5,-40($0)
730    bne a4,a5,.L4
731 .L177:
732    lw a4,-36($0)
733    lw a5,-40($0)
734    bne a4,a5,.L4
735 .L178:
736    lw a4,-36($0)
737    lw a5,-40($0)
738    bne a4,a5,.L4
739 .L179:
740    lw a4,-36($0)
741    lw a5,-40($0)
742    bne a4,a5,.L4
743 .L180:
744    lw a4,-36($0)
745    lw a5,-40($0)
746    bne a4,a5,.L4
747 .L181:
748    lw a4,-36($0)
749    lw a5,-40($0)
750    bne a4,a5,.L4
751 .L182:
752    lw a4,-36($0)
753    lw a5,-40($0)
754    bne a4,a5,.L4
755 .L183:
756    lw a4,-36($0)
757    lw a5,-40($0)
758    bne a4,a5,.L4
759 .L184:
760    lw a4,-36($0)
761    lw a5,-40($0)
762    bne a4,a5,.L4
763 .L185:
764    lw a4,-36($0)
765    lw a5,-40($0)
766    bne a4,a5,.L4
767 .L186:
768    lw a4,-36($0)
769    lw a5,-40($0)
770    bne a4,a5,.L4
771 .L187:
772    lw a4,-36($0)
773    lw a5,-40($0)
774    bne a4,a5,.L4
775 .L188:
776    lw a4,-36($0)
777    lw a5,-40($0)
778    bne a4,a5,.L4
779 .L189:
780    lw a4,-36($0)
781    lw a5,-40($0)
782    bne a4,a5,.L4
783 .L190:
784    lw a4,-36($0)
785    lw a5,-40($0)
786    bne a4,a5,.L4
787 .L191:
788    lw a4,-36($0)
789    lw a5,-40($0)
790    bne a4,a5,.L4
791 .L192:
792    lw a4,-36($0)
793    lw a5,-40($0)
794    bne a4,a5,.L4
795 .L193:
796    lw a4,-36($0)
797    lw a5,-40($0)
798    bne a4,a5,.L4
799 .L194:
800    lw a4,-36($0)
801    lw a5,-40($0)
802    bne a4,a5,.L4
803 .L195:
804    lw a4,-36($0)
805    lw a5,-40($0)
806    bne a4,a5,.L4
807 .L196:
808    lw a4,-36($0)
809    lw a5,-40($0)
810    bne a4,a5,.L4
811 .L197:
812    lw a4,-36($0)
813    lw a5,-40($0)
814    bne a4,a5,.L4
815 .L198:
816    lw a4,-36($0)
817    lw a5,-40($0)
818    bne a4,a5,.L4
819 .L199:
820    lw a4,-36($0)
821    lw a5,-40($0)
822    bne a4,a5,.L4
823 .L200:
824    lw a4,-36($0)
825    lw a5,-40($0)
826    bne a4,a5,.L4
827 .L201:
828    lw a4,-36($0)
829    lw a5,-40($0)
830    bne a4,a5,.L4
831 .L202:
832    lw a4,-36($0)
833    lw a5,-40($0)
834    bne a4,a5,.L4
835 .L203:
836    lw a4,-36($0)
837    lw a5,-40($0)
838    bne a4,a5,.L4
839 .L204:
840    lw a4,-36($0)
841    lw a5,-40($0)
842    bne a4,a5,.L4
843 .L205:
844    lw a4,-36($0)
845    lw a5,-40($0)
846    bne a4,a5,.L4
847 .L206:
848    lw a4,-36($0)
849    lw a5,-40($0)
850    bne a4,a5,.L4
851 .L207:
852    lw a4,-36($0)
853    lw a5,-40($0)
854    bne a4,a5,.L4
855 .L208:
856    lw a4,-36($0)
857    lw a5,-40($0)
858    bne a4,a5,.L4
859 .L209:
860    lw a4,-36($0)
861    lw a5,-40($0)
862    bne a4,a5,.L4
863 .L210:
864    lw a4,-36($0)
865    lw a5,-40($0)
866    bne a4,a5,.L4
867 .L211:
868    lw a4,-36($0)
869    lw a5,-40($0)
870    bne a4,a5,.L4
871 .L212:
872    lw a4,-36($0)
873    lw a5,-40($0)
874    bne a4,a5,.L4
875 .L213:
876    lw a4,-36($0)
877    lw a5,-40($0)
878    bne a4,a5,.L4
879 .L214:
880    lw a4,-36($0)
881    lw a5,-40($0)
882    bne a4,a5,.L4
883 .L215:
884    lw a4,-36($0)
885    lw a5,-40($0)
886    bne a4,a5,.L4
887 .L216:
888    lw a4,-36($0)
889    lw a5,-40($0)
890    bne a4,a5,.L4
891 .L217:

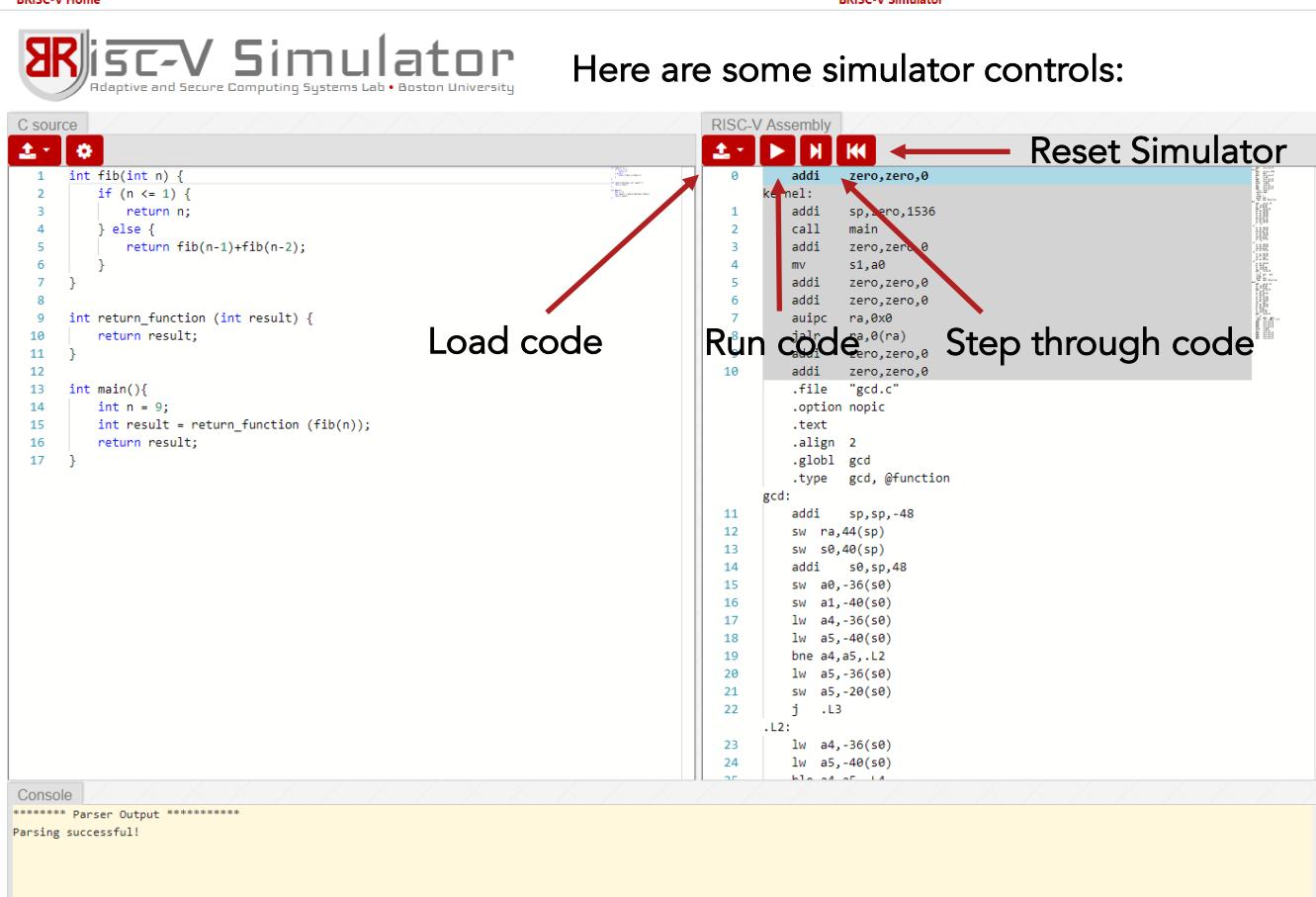
```

Simulator Controls

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

Here are some simulator controls:



Load code

Run code

Reset Simulator

Step through code

Registers **Memory**

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

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					0010011
000000000000	00000	000	00000						

Console
***** Parser Output *****
Parsing successful!

Stepping Through a Program

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```

1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }

```

Let's click the step button twice

RISC-V Assembly

```

0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
18 gcd:
19     addi sp,sp,-48
20     sw ra,44(sp)
21     sw s0,40(sp)
22     addi s0,sp,48
23     sw a0,-36(s0)
24     sw a1,-40(s0)
25     lw a4,-36(s0)
26     lw a5,-40(s0)
27     lw a5,-40(s0)
28     bne a4,a5,.L2
29     lw a5,-36(s0)
30     sw a5,-20(s0)
31     j .L3
32 .L2:
33     lw a4,-36(s0)
34     lw a5,-40(s0)
35     bne a4,a5,.L4

```

The blue line shows which instruction will be executed next

Registers **Memory**

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

Console

```
***** Parser Output *****
Parsing successful!
```

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					0010011
000000000000	00000	000	00000						

Stepping Through a Program

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

The blue line moved two lines down

BRISC-V Simulator

RISC-V Assembly

```
0 addi zero,zero,0
1 addi sp,zero,1536
2 call main
3 addi zero,zero,0
4 mv $1,a0
5 addi zero,zero,0
6 addi zero,zero,0
7 auipc ra,0x0
8 jalr ra,0(ra)
9 addi zero,zero,0
10 addi zero,zero,0
11 .file "gcd.c"
12 .option nopic
13 .text
14 .align 2
15 .globl gcd
16 .type gcd, @function
gcd:
17 addi sp,sp,-48
18 sw ra,44(sp)
19 sw $0,40(sp)
20 addi $0,sp,48
21 sw a0,-36($0)
22 sw a1,-40($0)
23 lw a4,-36($0)
24 lw a5,-40($0)
25 lw a5,-36($0)
26 sw a5,-20($0)
27 j .L3
.L2:
28 lw a4,-36($0)
29 lw a5,-40($0)
30 bne a4,a5,.L2
31 lw a4,-36($0)
32 lw a5,-40($0)
33 bne a4,a5,.L2
34 j .L3
```

Click the step button again

These instructions did not do anything, but the next one will

The instruction breakdown pane shows how the instruction is stored in memory

The top row represents the bit ranges, the middle row shows range names, the bottom row shows binary values for the specific instruction

Registers **Memory**

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

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm			rs1	funct3	rd		opcode		
011000000000	00000	000	00010	0010011					

addi Changed the Register File

BRISC-V Home

BRISC-V Simulator

Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

addi sp, zero, 1536
got executed

It summed 0 and 1536,
and put it in register sp

Console

```
***** Parser Output *****
Parsing successful!
```

BRISC-V Simulator

RISC-V Assembly

```
0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
18 gcd:
19     addi sp,sp,-48
20     sw ra,44(sp)
21     sw s0,40(sp)
22     addi s0,sp,48
23     sw a0,-36($0)
24     sw a1,-40($0)
25     lw a4,-36($0)
26     lw a5,-40($0)
27     lw a5,-40($0)
28     bne a4,a5,L2
29     lw a5,-36($0)
30     sw a5,-20($0)
31     j .L3
32 .L2:
33     lw a4,-36($0)
34     lw a5,-40($0)
35     bne a4,a5,L4
```

Register sp is highlighted!

sp stands for stack pointer

Manual & Examples

ASCS | ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Registers	Memory	Register	Value
zero	[0]	ra	[1]
sp	1536	gp	[3]
tp	0	t0	[5]
t1	0	t2	[7]
s0/p	0	s1	[9]
a0	0	a1	[11]
a2	0	a3	[13]
a4	0	a5	[15]
a6	0	a7	[17]
s2	0	s3	[19]
s4	0	s5	[21]
s6	0	s7	[23]
s8	0	s9	[25]
s10	0	s11	[27]
t3	0	t4	[29]
t5	0	t6	[31]

Instruction breakdown

Pseudo-Instructions don't have a breakdown

Setting Breakpoints

BRISC-V Home

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

Right-clicking on an instruction opens a menu

RISC-V Assembly

```
0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
18 gcd:
19     addi sp,sp,-48
20     sw ra,44(sp)
21     sw s0,40(sp)
22     addi s0,s0,-48
23     sw a0,-36($0)
24     sw a1,-36($0)
25     lw a0,-36($0)
26     lw a1,-36($0)
27     bne a4,a5,.L2
28     lw a5,-40($0)
29     lw a5,-40($0)
30     bne a4,a5,.L2
31     lw a5,-36($0)
32     sw a5,-20($0)
33     j .L3
34
35 .L2:
36     lw a4,-36($0)
37     lw a5,-40($0)
38     bne a4,a5,.L2
```

Console

```
***** Parser Output *****
Parsing successful!
```

BRISC-V Simulator

Manual & Examples

ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Registers	Memory		
Register	Value	Register	Value
zero	[0] 0	ra	[1] 0
sp	[2] 1536	gp	[3] 0
tp	[4] 0	t0	[5] 0
t1	[6] 0	t2	[7] 0
s0/fp	[8] 0	s1	[9] 0
a0	[10] 0	a1	[11] 0
a2	[12] 0	a3	[13] 0
a4	[14] 0	a5	[15] 0
a6	[16] 0	a7	[17] 0
s2	[18] 0	s3	[19] 0
s4	[20] 0	s5	[21] 0
s6	[22] 0	s7	[23] 0
s8	[24] 0	s9	[25] 0
s10	[26] 0	s11	[27] 0
t3	[28] 0	t4	[29] 0
t5	[30] 0	t6	[31] 0

Registers

Memory

Instruction breakdown

Pseudo-Instructions don't have a breakdown

Add breakpoint Ctrl+B

Delete breakpoint Ctrl+D

Copy

Command Palette F1

Let's click on the first item – Add breakpoint

Setting Breakpoints

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

Now if we click run, simulator will keep executing instructions until it hits a breakpoint

A breakpoint got created!

RISC-V Assembly

```
0 addi zero,zero,0
1 kernel:
2 addi sp,zero,1536
3 call main
4 addi zero,zero,0
5 mv s1,a0
6 addi zero,zero,0
7 addi zero,zero,0
8 auipc ra,0x0
9 jalr ra,0(ra)
10 addi zero,zero,0
11 addi zero,zero,0
12 .file "gcd.c"
13 .option nopic
14 .text
15 .align 2
16 .globl gcd
17 .type gcd, @function
18 gcd:
19     addi sp,sp,-48
20     sw ra,44(sp)
21     sw s0,40(sp)
22     addi s0,sp,48
23     sw a0,-36(s0)
24     sw a1,-40(s0)
25     lw a4,-36(s0)
26     lw a5,-40(s0)
27     bne a4,a5,.L2
28     lw a5,-36(s0)
29     sw a5,-20(s0)
30     j .L3
31 .L2:
32     lw a4,-36(s0)
33     lw a5,-40(s0)
34     bne a4,a5,.L4
```

Registers Memory

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

Instruction breakdown

Pseudo-Instructions don't have a breakdown

Console

```
***** Parser Output *****
Parsing successful!
```

asclab.org/research/briscv/index.html

Running Code until a Breakpoint

BRISC-V Home

BRISC-V Simulator

ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

C source

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }
```

RISC-V Assembly

```
0 addi zero,zero,0
kernel:
1 addi sp,zero,1536
2 call main
3 addi zero,zero,0
4 mv s1,a0
5 addi zero,zero,0
6 addi zero,zero,0
7 auipc ra,0x0
8 jalr ra,(ra)
9 addi zero,zero,0
10 addi zero,zero,0
.file "gcd.c"
.option nopic
.text
.align 2
.globl gcd
.type gcd, @function
gcd:
11 addi sp,sp,-48
12 sw ra,44(sp)
13 sw s0,40(sp)
14 addi s0,sp,48
15 sw a0,-36($0)
16 sw a1,-40($0)
17 lw a4,-36($0)
18 lw a5,-40($0)
19 bne a4,a5,.L2
20 lw a5,-36($0)
21 sw a5,-20($0)
22 j .L3
.L2:
23 lw a4,-36($0)
24 lw a5,-40($0)
.L3:
25 lw a4,-36($0)
26 lw a5,-40($0)
27 bne a4,a5,.L2
28 lw a5,-36($0)
29 sw a5,-20($0)
30 j .L3
.L4:
31 lw a4,-36($0)
32 lw a5,-40($0)
33 bne a4,a5,.L4
34 lw a5,-36($0)
35 sw a5,-20($0)
36 j .L4
```

Registers

Register	Value	Register	Value
zero	[0] 0	ra	[1] 73
sp	[2] 1456	gp	[3] 0
tp	[4] 0	t0	[5] 0
t1	[6] 0	t2	[7] 0
s0/fp	[8] 1536	s1	[9] 0
a0	[10] 64	a1	[11] 48
a2	[12] 0	a3	[13] 0
a4	[14] 0	a5	[15] 48
a6	[16] 0	a7	[17] 0
s2	[18] 0	s3	[19] 0
s4	[20] 0	s5	[21] 0
s6	[22] 0	s7	[23] 0
s8	[24] 0	s9	[25] 0
s10	[26] 0	s11	[27] 0
t3	[28] 0	t4	[29] 0
t5	[30] 0	t6	[31] 0

Console

```
***** Parser Output *****
Parsing successful!
```

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					
000000110000	00010	000	01000	0010011					

The instruction pointer
moved to the breakpoint!

A lot of registers changed
values!

Text and Data Sections

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```

1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8 int result = fib(9);
9 int main(){
10     int n = 9;
11     int result = return_function (fib(n));
12     return result;
13 }
14
15 }
```

RISC-V Assembly

```

22      ecall
# let's statically allocate a string "HELLO!"
# we start this by creating a read-only data section
23      .rodata
24      .HELLO:
# strings should end with the null terminator \0
# the null terminator's binary value is 0!
# we split HELLO\0 into two 32bit words:
# HELL and 0!00 - note that that's an "0!" and 2 zeros
# we write HELLO in ascii:
# H - 0x48
# E - 0x45
# L - 0x4C
# since this is a little-endian architecture, we
# write HELLO in reverse - LEHH
25      .word 0x4C4C4548
# now we write the second part 0!00
26      .word 0x0000214F
# this section is parsed when you load the program -
# not when the instruction pointer runs over it.
# as soon as you loaded the program, you should see
# this string in the memory pane's data section,
# somewhere close to the bottom of it.
#
# now we can go back to a text section that has code
27      .text
# print the string "HELLO!\n"
addi t0, zero, 3      # this is the string printing syscall
lui a0, %hi(.HELLO)  # this loads the top 20 bits
# of .HELLO address into a0
28      addi a0, a0, %lo(.HELLO) # this loads the bottom 12 bits
addi a1, zero, 7      # length of the string
ecall
# print characters '!', '\n', '-'
addi t0, zero, 3

```

Registers **Memory**

REG	HEX	DEC	BINARY
0x00000138	00 00 00 00	0	0000000000000000
0x00000134	00 00 00 00	0	0000000000000000
0x00000130	00 00 00 00	0	0000000000000000
0x0000012C	00 00 00 00	0	0000000000000000
0x00000128	00 00 00 00	0	0000000000000000
0x00000124	00 00 00 00	0	0000000000000000
0x00000120	00 00 00 00	0	0000000000000000
0x0000011C	00 00 00 00	0	0000000000000000
0x00000118	00 00 00 00	0	0000000000000000
0x00000114	00 00 00 00	0	0000000000000000
0x00000110	00 00 00 00	0	0000000000000000
0x0000010C	00 00 00 00	0	0000000000000000
HEAP SEGMENT			
0x00000108	00 00 21 4f	1000000000000000	0000000000000000
0x00000104	4c 45 4f	4000000000000000	// <-- .hello
TEXT SEGMENT			
0x00000100	00 00 00 13	1000000000000000	// addi zero,zero
0x000000fc	00 00 00 13	1000000000000000	// addi zero,zero
0x000000f8	00 00 00 13	1000000000000000	// addi zero,zero
0x000000f4	00 00 00 13	1000000000000000	// addi zero,zero
0x000000f0	00 00 80 e7	1000000000000000	// jalr ra,0(ra)
0x000000ec	00 00 00 97	1000000000000000	// auipc ra,0x0
0x000000e8	00 00 00 13	1000000000000000	// addi zero,zero
0x000000e4	00 00 00 13	1000000000000000	// addi zero,zero
0x000000e0	00 00 00 13	1000000000000000	// addi zero,zero
0x000000dc	00 00 00 13	1000000000000000	// addi zero,zero
0x000000d8	00 00 80 67	1000000000000000	// jr ra
0x000000d4	00 00 00 73	1000000000000000	// ecall
0x000000d0	71 00 05 13	1000000000000000	// addi a0, zero,
0x000000cc	00 70 02 93	1000000000000000	// addi t0, zero,
0x000000c8	00 00 00 73	1000000000000000	// ecall
0x000000c4	ff 00 05 13	1000000000000000	// addi a0, zero,
0x000000c0	00 70 02 93	1000000000000000	// addi t0, zero,
0x000000b8	00 00 00 73	1000000000000000	// ecall

Console

```
***** Parser Output *****
Parsing successful!
```

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					
0000000000000000	00000	000	00000	0010011					

Text and Data Sections

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```

1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8 int result = return_function(fib(9));
9 return result;
10 }
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }

```

Labels in data sections point to memory statically allocated after them

Here we allocate the string "HELLO!" by into two 32bit words – one with "HELL" and one with "O!00"

RISC-V Assembly

```

22      ecall
# let's statically allocate a string "HELLO!"
# we start this by creating a read-only data section
.rodata
.HELLO:
# strings should end with the null terminator \0
# the null terminator's binary value is 0!
# we split HELLO\0 into two 32bit words:
# HELL and O!00 - note that that's an "0!" and 2 zeros
# we write HELLO in ascii:
# H - 0x48
# E - 0x45
# L - 0x4C
# since this is a little-endian architecture, we
# write HELL in reverse - LEHH
.word 0x4C4C4548
# now we write the second part O!00
.word 0x0000214F
# this section is parsed when you load the program -
# not when the instruction pointer runs over it.
# as soon as you loaded the program, you should see
# this string in the memory pane's data section,
# somewhere close to the bottom of it.
#
# now we can go back to a text section that has code
.text
# print the string "HELLO!\n"
addi t0, zero, 3      # this is the string printing syscall
lui a0, %hi(.HELLO)  # this loads the top 20 bits
                      # of .HELLO address into a0
addi a0, a0, %lo(.HELLO) # this loads the bottom 12 bits
addi a1, zero, 7      # length of the string
ecall
# print characters '!', '\n', '-'
addi t0, zero, 3

```

Registers

REG	HEX	DEC	BINARY
0	0x00000138	00 00 00 00	0000000000000000
1	0x00000134	00 00 00 00	0000000000000000
2	0x00000130	00 00 00 00	0000000000000000
3	0x0000012C	00 00 00 00	0000000000000000
4	0x00000128	00 00 00 00	0000000000000000
5	0x00000124	00 00 00 00	0000000000000000
6	0x00000120	00 00 00 00	0000000000000000
7	0x0000011C	00 00 00 00	0000000000000000
8	0x00000118	00 00 00 00	0000000000000000
9	0x00000114	00 00 00 00	0000000000000000
10	0x00000110	00 00 00 00	0000000000000000
11	0x0000010C	00 00 00 00	0000000000000000
12			
13			
14			
15			
16			
17			

Memory

ADDR	HEX	DEC	BINARY
0x00000108	00 00 21 4f	00 00 21 4f	0000000000000000
0x00000104	4c 45 4f 48	// <- .hello	0000000000000000
0x00000100	00 00 00 13	// addi zero,zero	0000000000000000
0x000000fc	00 00 00 13	// addi zero,zero	0000000000000000
0x000000f8	00 00 00 13	// addi zero,zero	0000000000000000
0x000000f4	00 00 00 13	// addi zero,zero	0000000000000000
0x000000f0	00 00 88 e7	// jalr ra,0(ra)	0000000000000000
0x000000ec	00 00 00 97	// auipc ra,0x0	0000000000000000
0x000000e8	00 00 00 13	// addi zero,zero	0000000000000000
0x000000e4	00 00 00 13	// addi zero,zero	0000000000000000
0x000000e0	00 00 00 13	// addi zero,zero	0000000000000000
0x000000dc	00 00 00 13	// addi zero,zero	0000000000000000
0x000000d8	00 00 88 67	// jr ra	0000000000000000
0x000000d4	00 00 00 73	// ecall	0000000000000000
0x000000d0	71 00 05 13	// addi a0, zero,	0000000000000000
0x000000cc	00 70 02 93	// addi t0, zero,	0000000000000000
0x000000c8	00 00 00 73	// ecall	0000000000000000
0x000000c4	ff 00 05 13	// addi a0, zero,	0000000000000000
0x000000c0	00 70 02 93	// addi t0, zero,	0000000000000000
0x000000b8	00 00 00 73	// ecall	0000000000000000

Console

```
***** Parser Output *****
Parsing successful!
```

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					
0000000000000000	00000	000	00000	0010011					

Text and Data Sections

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

C source

```

1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     } else {
5         return fib(n-1)+fib(n-2);
6     }
7 }
8
9 int return_function (int result) {
10    return result;
11 }
12
13 int main(){
14     int n = 9;
15     int result = return_function (fib(n));
16     return result;
17 }

```

RISC-V Assembly

```

22      ecall
# let's statically allocate a string "HELLO!"
# we start this by creating a read-only data section
.rodata
.HELLO:
# strings should end with the null terminator \0
# the null terminator's binary value is 0!
# we split HELLO\0 into two 32bit words:
# HELL and 0!00 - note that that's an "0!" and 2 zeros
# we write HELLO in ascii:
# H - 0x48
# E - 0x45
# L - 0x4C
# since this is a little-endian architecture, we
# write HELLO in reverse - LEHH
.word 0x4C4C4548
# now we write the second part 0!00
.word 0x0000214F
# this section is parsed when you load the program -
# not when the instruction pointer runs over it.
# as soon as you load the program, you should see
# this string in the memory pane's data section,
# somewhere close to the bottom of it.
#
# now we can go back to a text section that has code
.text
# print the string "HELLO!\n"
addi t0, zero, 3      # this is the string printing syscall
lui a0, %hi(.HELLO)  # this loads the top 20 bits
                      # of .HELLO address into a0
addi a0, a0, %lo(.HELLO) # this loads the bottom 12 bits
addi a1, zero, 7      # length of the string
ecall
# print characters '!', '\n', '-'
addi t0, zero, 3

```

Registers **Memory**

HEX	DEC	BINARY
0x00000138:	00 00 00 00	
0x00000134:	00 00 00 00	
0x00000130:	00 00 00 00	
0x0000012C:	00 00 00 00	
0x00000128:	00 00 00 00	
0x00000124:	00 00 00 00	
0x00000120:	00 00 00 00	
0x0000011C:	00 00 00 00	
0x00000118:	00 00 00 00	
0x00000114:	00 00 00 00	
0x00000110:	00 00 00 00	
0x0000010C:	00 00 00 00	
HEAP SEGMENT		
DATA SEGMENT		
0x00000108:	00 00 21 4f	
0x00000104:	4c 4c 45 48	// <- .hello
TEXT SEGMENT		
0x00000100:	00 00 00 13	// addi zero,zero
0x000000fc:	00 00 00 13	// addi zero,zero
0x000000f8:	00 00 00 13	// addi zero,zero
0x000000f4:	00 00 00 13	// addi zero,zero
0x000000f0:	00 00 88 e7	// jalr ra,0(ra)
0x000000ec:	00 00 00 97	// auipc ra,0x0
0x000000e8:	00 00 00 13	// addi zero,zero
0x000000e4:	00 00 00 13	// addi zero,zero
0x000000e0:	00 00 00 13	// addi zero,zero
0x000000dc:	00 00 00 13	// addi zero,zero
0x000000d8:	00 00 88 67	// jr ra
0x000000d4:	00 00 00 73	// ecall
0x000000d0:	71 00 05 13	// addi a0, zero,
0x000000cc:	00 70 02 93	// addi t0, zero,
0x000000c8:	00 00 00 73	// ecall
0x000000c4:	ff 00 05 13	// addi a0, zero,
0x000000c0:	00 70 02 93	// addi t0, zero,
0x000000b8:	00 00 00 73	// ecall

Console

```
***** Parser Output *****
Parsing successful!
```

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					
000000000000	00000	000	00000	0010011					

System Calls

- We also provide some simple system calls
- System calls are used for functionalities provided by the operating system
 - Think file systems, IO, etc.
- In RISC-V, system calls look something like:
 - Put the type of system call you want in register t0
 - More about that on the next slide
 - Put any arguments you may have in a0 and a1
 - Call instruction ECALL
 - If the system call has return values, they will be in a0
- **To really get familiar with syscalls, try running the example syscall file in the simulator**

Supported Syscalls

Syscall	Syscall ID (put this in t0)	Description
Print integer	1	Print integer value in a0 to console
Print char	2	Print ascii value in a0 to console
Print string	3	Print string with address in a0 and length in a1 to console
Read integer	4	Read integer from console into a0
Read char	5	Read character from console into a0 as an ascii value
Read string	6	Read string of length given in a1 from console and store it at address in a0
SBRK	7	Dynamically allocate the amount of bytes specified in a0. The pointer to the beginning of the newly allocated memory will be stored in a0. The value in a0 can be negative, if you want to deallocate some memory!

BRISC-V Home BRISC-V Simulator Manual & Examples

BRISC-V Simulator
Adaptive and Secure Computing Systems Lab • Boston University

RISC-V Assembly

```
0  addi zero,zero,0
1  kernel:
2  addi sp,zero,1536
3  call main
4  addi zero,zero,0
5  mv s0,a0
6  addi zero,zero,0
7  addi zero,zero,0
8  addi zero,zero,0
9  addi zero,zero,0
10 addi zero,zero,0
11 .file "gcd.c"
12 .option nopic
13 .text
14 .align 2
15 .globl gcd
16 .type gcd, @function
17
18
19
20
21
22
23
24
```

Registers **Memory**

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

Console

```
***** Parser Output *****
Parsing successful!
```

Instruction breakdown

31	20	19	15	14	12	11	7	6	0
imm	rs1	funct3	rd	opcode					
000000110000	00010	000	01000	0010011					

That's All Folks!
Now open a text editor,
write some assembly,
and try to run it!