

Introduction to Computer Architecture

RISC-V Assembler and Runtime Simulator



R. Pacalet

Telecom Paris / EURECOM

The Graphical User Interface

The screenshot displays the RISC-V Assembler and Runtime Simulator GUI. The interface includes a menu bar (File, Edit, Run, Settings, Tools, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom.

Text Segment: This panel shows the assembly code being executed. The columns are Bkpt, Address, Code, Basic, and Source. The code includes instructions like `auipc x10, 0x000fc10`, `addi x10, x10, 0`, `addi x17, x0, 4`, `ecall`, `lui x8, 0xfffffff0`, `addi x8, x8, 0`, `lui x9, 0xfffffff0`, `addi x9, x9, 4`, `lw x5, 0(x8)`, `andi x5, x5, 1`, `beq x5, x0, 0xfffffff8`, and `li x5, 0(x5)`.

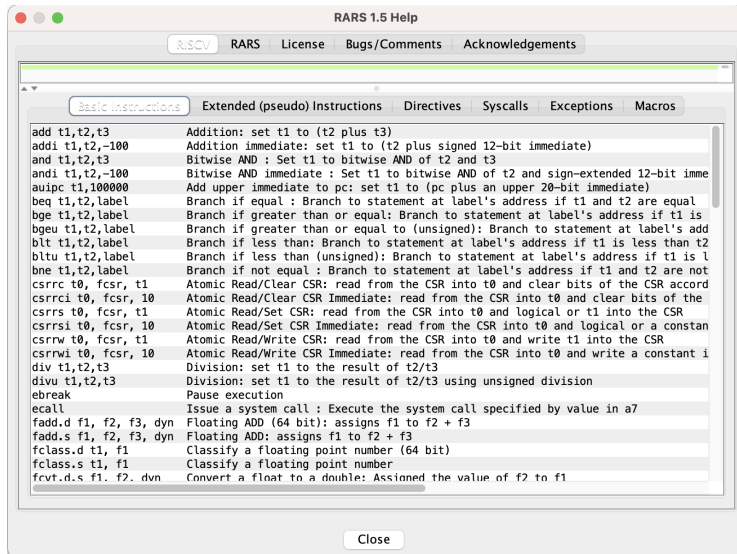
Labels: This panel shows the labels defined in the code. The columns are Label, Address, and Value. Labels include `(global)`, `main`, `code.s`, `main_get`, `main_end`, `hw`, `yt`, and `bye`.

Data Segment: This panel shows the data segment. The columns are Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c). The data is represented as hexadecimal values.

Registers: This panel shows the registers. The columns are Name, Number, and Value. Registers include `zero`, `ra`, `sp`, `gp`, `tp`, `t0`, `t1`, `t2`, `s0`, `s1`, `a0`, `a1`, `a2`, `a3`, `a4`, `a5`, `a6`, `a7`, `s2`, `s3`, `s4`, `s5`, `s6`, `s7`, `s8`, `s9`, `s10`, `s11`, `t3`, `t4`, `t5`, `t6`, and `pc`.

Messages: This panel shows the output of the program. The messages are: "Hello world!", "You typed: 4", "You typed: 2", "Bye!", and "-- program is finished running (0) --".

The vital Help window



Memory layout

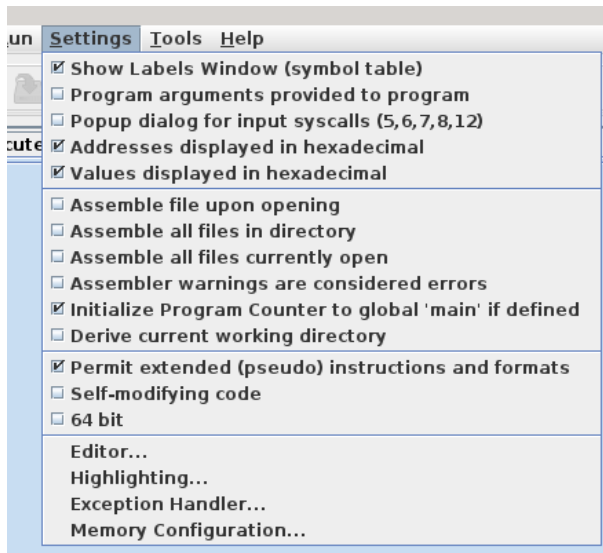
Memory Configuration

Configuration

- ☒ Default
- ☐ Compact, Data at Address 0
- ☐ Compact, Text at Address 0

0xffffffff	memory map limit address
0xffffffff	kernel space high address
0xffff0000	MMIO base address
0x80000000	kernel space base address
0x7fffffff	user space high address
0x7fffffff	data segment limit address
0x7fffffcc	stack base address
0x7ffffefc	stack pointer (sp)
0x10040000	stack limit address
0x10040000	heap base address
0x10010000	.data base address
0x10008000	global pointer (gp)
0x10000000	data segment base address
0x10000000	.extern base address
0x0ffffffc	text limit address
0x00400000	.text base address

Apply and Close Apply Cancel Reset



Directives, labels, pseudo instructions

```
1 .data
2 hw:
3 .asciz "Hello world!\n"
4
5 .text
6 .globl main
7 main:
8     la      a0,hw
9     li      a7,4
10    ecall
11    li      s0,0xffff0000
12    li      s1,0xffff0004
13 main_get:
14    lw      t0,0(s0)
15    andi    t0,t0,1
16    beq     t0,zero,main_get
```

Directives

- `.data`, `.text`, `.section`: segment change
- `.asciz`, `.ascii`, `.word`: strings, aligned words...
- `.globl label`: global label

Labels

Aliases for addresses

Pseudo instructions

- `la reg,label`: `lui (auipc) + addi`
- `li reg,imm`: `lui (auipc) + addi`

System calls (syscalls)

```
1 .data
2 hw:
3 .asciz "Hello world!\n"
4
5 .text
6 .globl main
7 main:
8     la      a0,hw
9     li      a7,4
10    ecall
11    li      s0,0xffff0000
12    li      s1,0xffff0004
13 main_get:
14    lw      t0,0(s0)
15    andi    t0,t0,1
16    beq     t0,zero,main_get
```

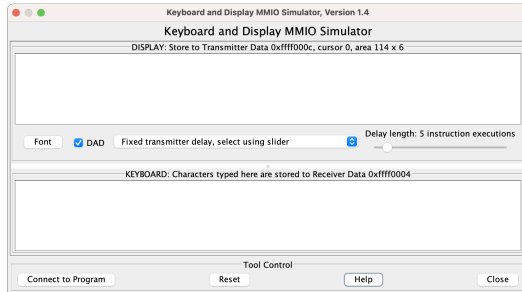
■ Access to system services

- Syscall ID in **a7**
- I/O in **a0-a6**
- **ecall**

■ Examples

- PrintInt (1), PrintString (4), PrintChar (11)
- ReadInt (5), ReadString (8)
- Exit (10), Sleep (32)
- RandSeed (40), RandInt (41)
- ... (more than 40)

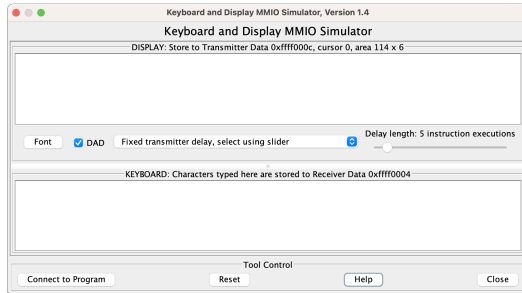
Memory Mapped IO: companion tool emulates keyboard and console



■ Keyboard (input)

- Ready bit: bit 0 of `mem[0xffff0000]`
- ASCII code of input character: least significant byte of `mem[0xffff0004]`
- Reading `mem[0xffff0004]` clears ready bit

Memory Mapped IO: companion tool emulates keyboard and console



- Console (display) with tunable delay (in # of executed instructions)
 - Ready bit: bit 0 of `mem[0xffff0008]`
 - ASCII code of character to display: write least significant byte of `mem[0xffff000c]`
 - Special features (ASCII art)

Other features

Companion tools

- Branch prediction simulator
- Data cache simulator
- Bitmap display

Custom exception handler

- Interrupt service routines
- Could be used to design a simple OS

Audio system calls

- MIDI outputs
- Could be used to play music (we will)