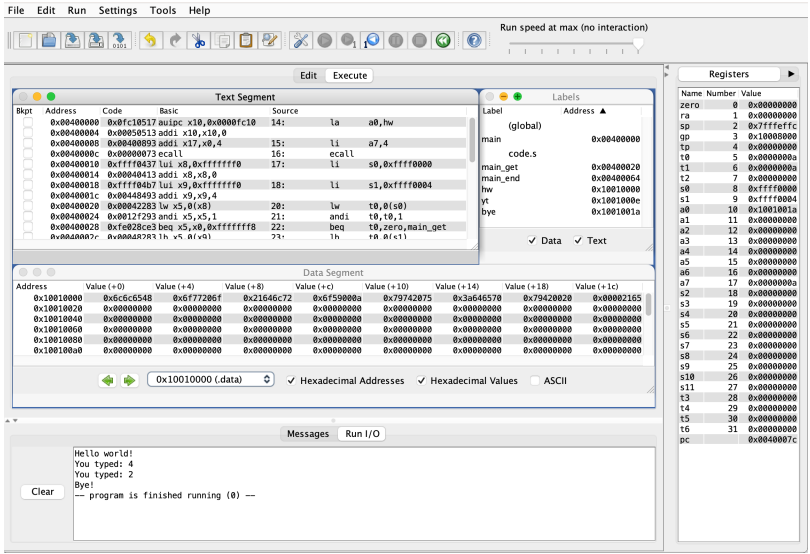
Introduction to Computer Architecture

RISC-V Assembler and Runtime Simulator

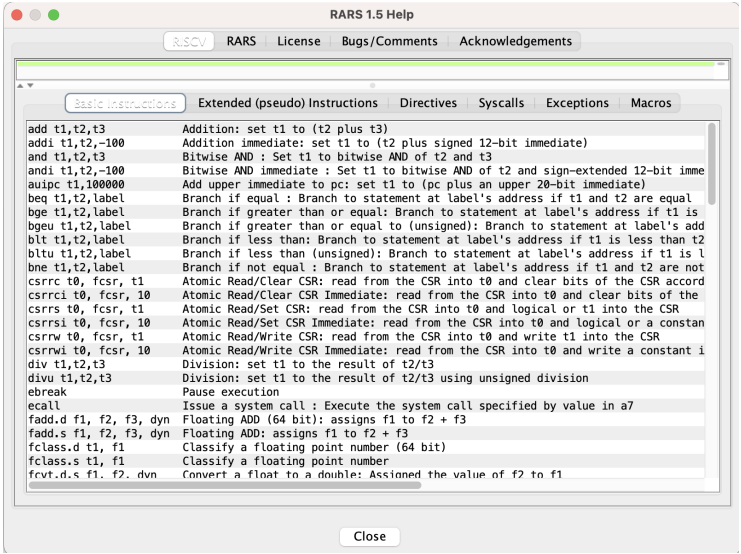
R. Pacalet

Telecom Paris / EURECOM

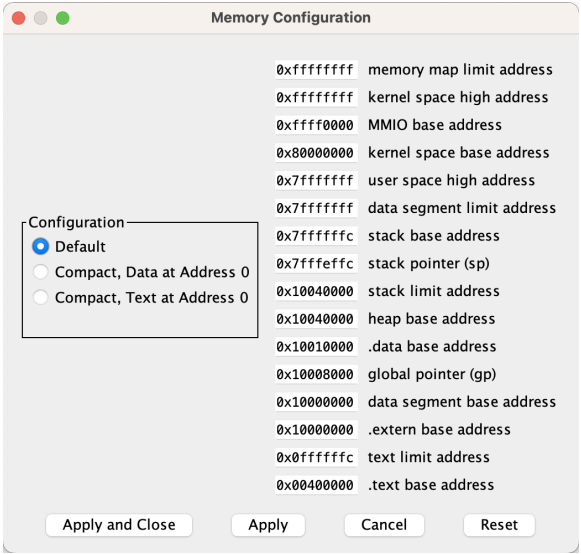
The Graphical User Interface



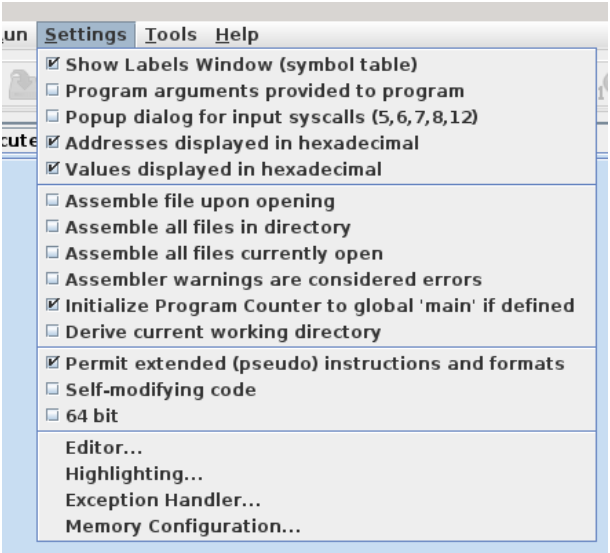
The vital Help window



Memory layout



Settings


un
Settings
Tools
Help
Show Labels Window (symbol table)
Program arguments provided to program
Popup dialog for input syscalls (5,6,7,8,12)
Addresses displayed in hexadecimal
Values displayed in hexadecimal
Assemble file upon opening
Assemble all files in directory
Assemble all files currently open
Assembler warnings are considered errors
Initialize Program Counter to global 'main' if defined
Derive current working directory
Permit extended (pseudo) instructions and formats
Self-modifying code
64 bit
Editor...
Highlighting...
Exception Handler...
Memory Configuration...

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

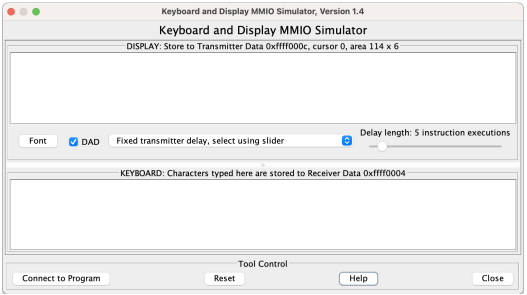
- `la reg,label`: `lui` (`auipc`) + `addi`
- `li reg,imm`: `lui` (`auipc`) + `addi`

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
```

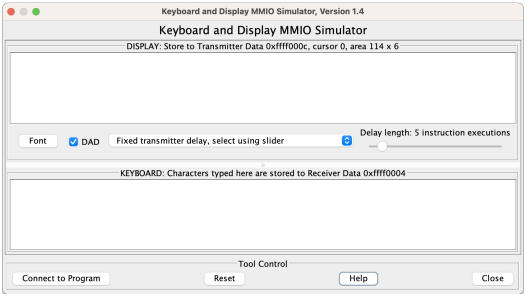
- Access to system services
  - Syscall ID in a7
  - I/O in a0–a6
  - ecall
- Examples
  - PrintInt (1), PrintString (4), PrintChar (11)
  - ReadInt (5), ReadString (8)
  - Exit (10), Sleep (32)
  - RandSeed (40), RandInt (41)
  - . . . (more than 40)

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)