RV Day 2 - Introduction to ABI and basic verification flow

RV-D2SK1 - Application Binary interface (ABI)

RV_D2SK1_L1_Introduction to Application Binary Interface

- Application -- (java, C, C++ interface) -- Program-Standard Libraries -- Operating System -- (ISA) -- RISC-V,ARM, x86 – RTL – Hardware.
- Available to programmer directly known as User ISA and User & system ISA.
- Application Program has a direct access the registers of RISC_V architecture via System call .
- The way the System call does is called as Application Binary Interface (System Call Interface).

RV_D2SK1_L2_Memory Allocation For Double Words

- Number can be loaded to registers in two different ways:
- First one is directly to 64bit registers, limited amount of registers so limited amount of data is stored. Secondly, major amount of data can be stored in the memory, and then from the memory it can be loaded to the register.
- RISC-V belongs to little-endian memory addressing system.

RV_D2SK1_L3_Load, Add And Store Instructions With Example

- Load Instructions & their representation in computer : -
  - Array M of 3 doubleword [ld   x8, 16(x23)]
  - size [32 bit]

0-6	opcode	load doubleword	ld
7-11	rd	destination register ‘rd’	x8
12-14	funct3	additional opcode bits	ld
15-19	rs1	source register ,5 bits	x23
20-31	immediate	offset ‘imm’	16

- Add Instructions: [add x8, x24, x8]

0-6	opcode	add command	add
7-11	rd	destination register	x8
12-14	funct3	additional opcode bits	add
15-19	rs1	source register ‘rs1’	x24
20-24	rs2	source register ‘rs2’	x8
25-31	funct7	additional opcode	add

- Store instruction (store back to memory) : [sd   x8, 8(x23)]

0-6	opcode	store doubleword	sd
7-11	immediate [4:0]	offset ‘imm’	8
12-14	funct3	additional opcode bits	sd
15-19	rs1	source register	x23
20-24	rs2	data register ‘rs2’	x8
25-31	immediate [11:5]	offset ‘imm’	8

RV_D2SK1_L4_Concluding 32-registers And Their Respective ABI Names

- R-type instruction: instructions which works only on registers, example add.
- I-type instruction: instructions which works on both registers and immediate registers , example load.
- S-type instructions: instructions which works on store/ source registers & the immediate registers also use to store registers, example store.

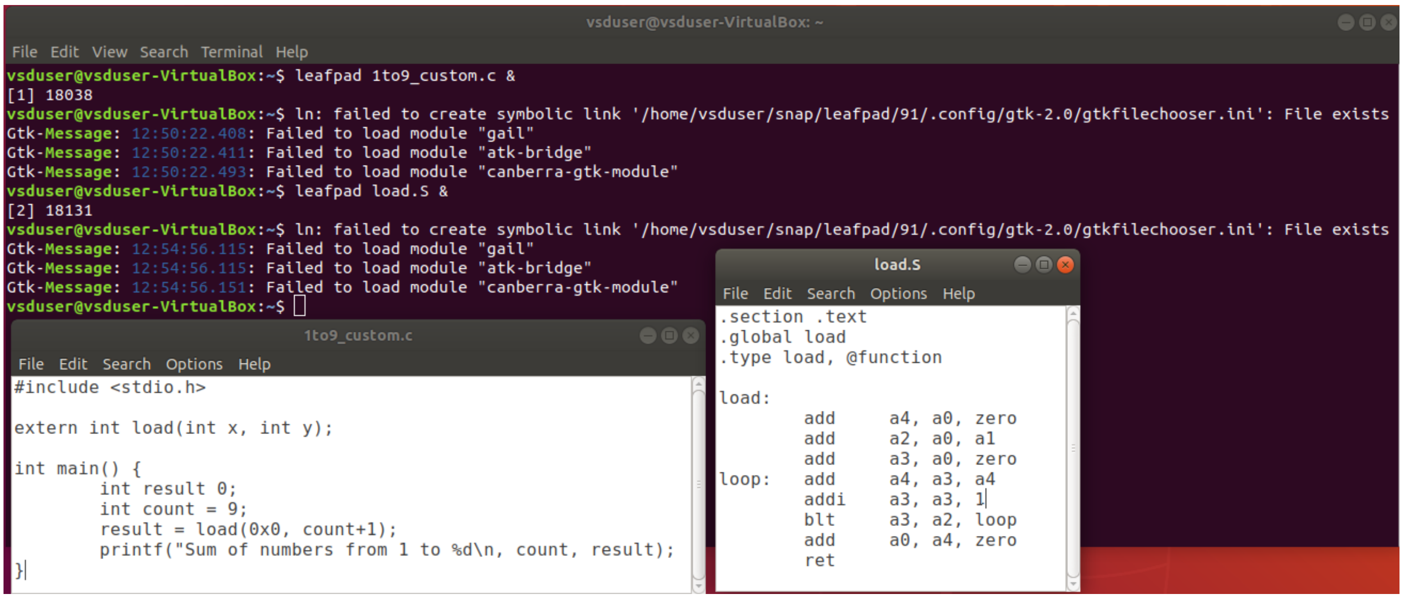
Register	ABI names	Usage	saver
X0	zero	hard-wired zero	-
X1	ra	return address	caller
X2	sp	stack pointers	callee
X3	gp	global pointers	-
X4	tp	thread pointers	-
X5-x7	t0-2	temporaries	caller
X8	s0/fp	saved registers/frame pointers	callee
X9	s1	saved registers	callee
X10-x11	a0-1	function arguments/return values	caller
X12-x17	a2-7	function arguments	caller
X18-x27	s2-11	saved registers	callee
X28-x31	t3-6	temporaries	callee

RV-D2SK2 - Lab work using ABI function calls

RV_D2SK2_L1_Study New Algorithm For Sum 1 to N Using ASM language

```
Start
Initialize a4 with ‘zero’
Initialize a3 with ‘zero’
Store count ’10 in a2
A4=a3+a4
A3=a3+1
Decision of whether a3<a2?
If ‘YES’ -- back to a4=a3+a4
If ‘NO’ -- A0=a4+zero
End
```

RV_D2SK2_L2_Review ASM Function Call

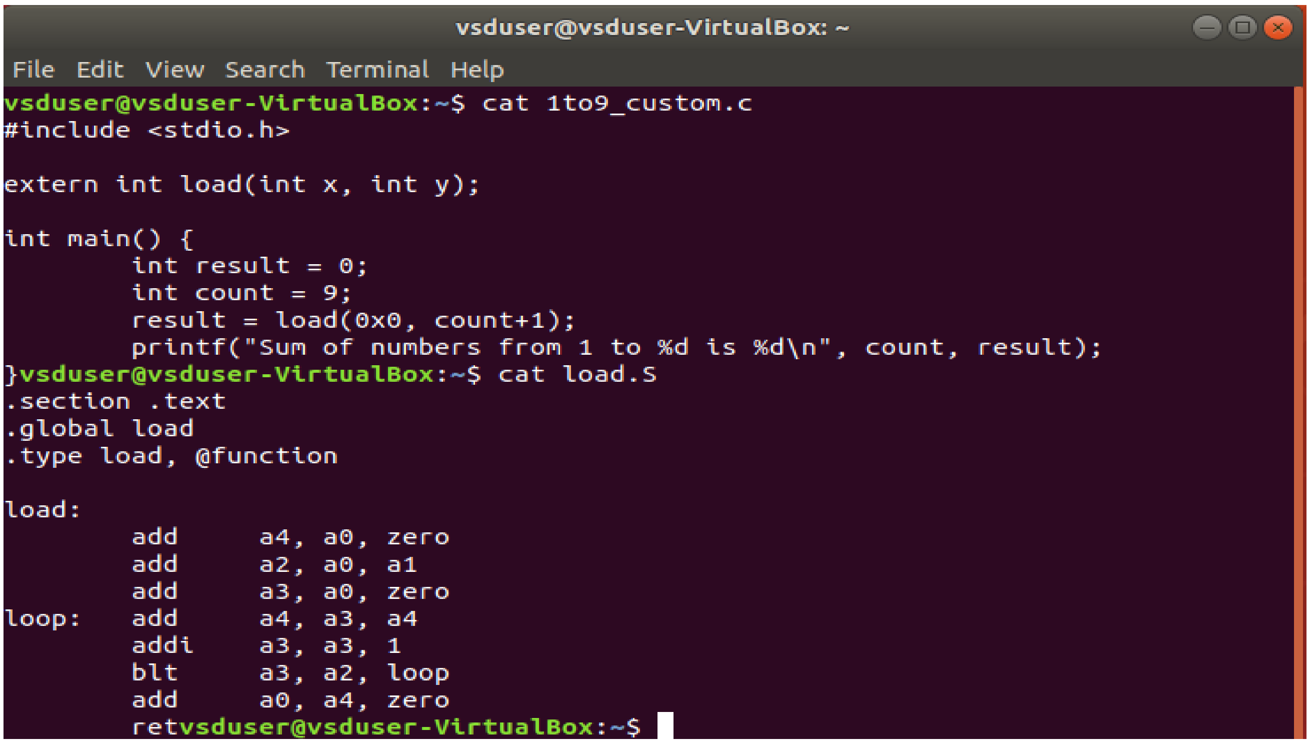

vsduser@vsduser-VirtualBox: ~
File Edit View Search Terminal Help
1to9_custom.c
File Edit Search Options Help
load.S
File Edit Search Options Help

```
vsduser@vsduser-VirtualBox:~$ leafpad 1to9_custom.c &
[1] 18038
vsduser@vsduser-VirtualBox:~$ ln: failed to create symbolic link '/home/vsduser/snap/leafpad/91/.config/gtk-2.0/gtkfilechooser.ini': File exists
Gtk-Message: 12:50:22.408: Failed to load module "gail"
Gtk-Message: 12:50:22.411: Failed to load module "atk-bridge"
Gtk-Message: 12:50:22.493: Failed to load module "canberra-gtk-module"
vsduser@vsduser-VirtualBox:~$ leafpad load.S &
[2] 18131
vsduser@vsduser-VirtualBox:~$ ln: failed to create symbolic link '/home/vsduser/snap/leafpad/91/.config/gtk-2.0/gtkfilechooser.ini': File exists
Gtk-Message: 12:54:56.115: Failed to load module "gail"
Gtk-Message: 12:54:56.115: Failed to load module "atk-bridge"
Gtk-Message: 12:54:56.151: Failed to load module "canberra-gtk-module"
vsduser@vsduser-VirtualBox:~$
```

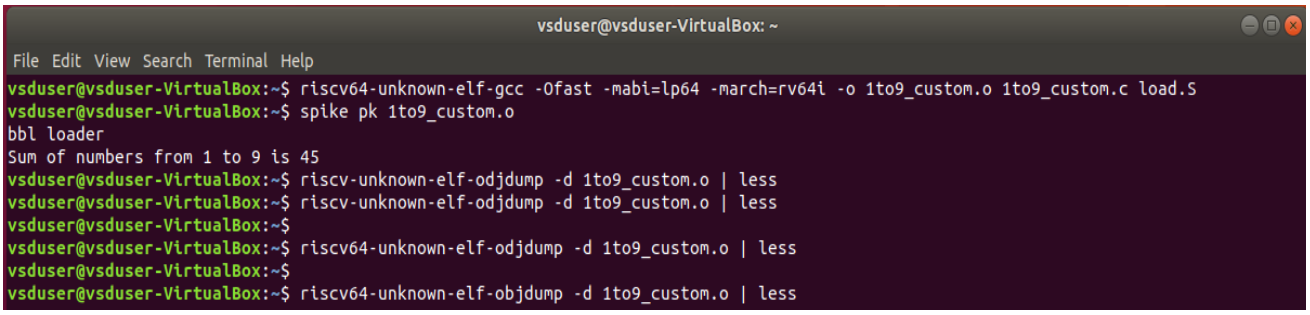
```
#include <stdio.h>

extern int load(int x, int y);

int main() {
        int result 0;
        int count = 9;
        result = load(0x0, count+1);
        printf("Sum of numbers from 1 to %d\n, count, result);
}
```

```
.section .text
.global load
.type load, @function

load:
        add     a4, a0, zero
        add     a2, a0, a1
        add     a3, a0, zero
loop:   add     a4, a3, a4
        addi    a3, a3, 1
        blt     a3, a2, loop
        add     a0, a4, zero
        ret
```

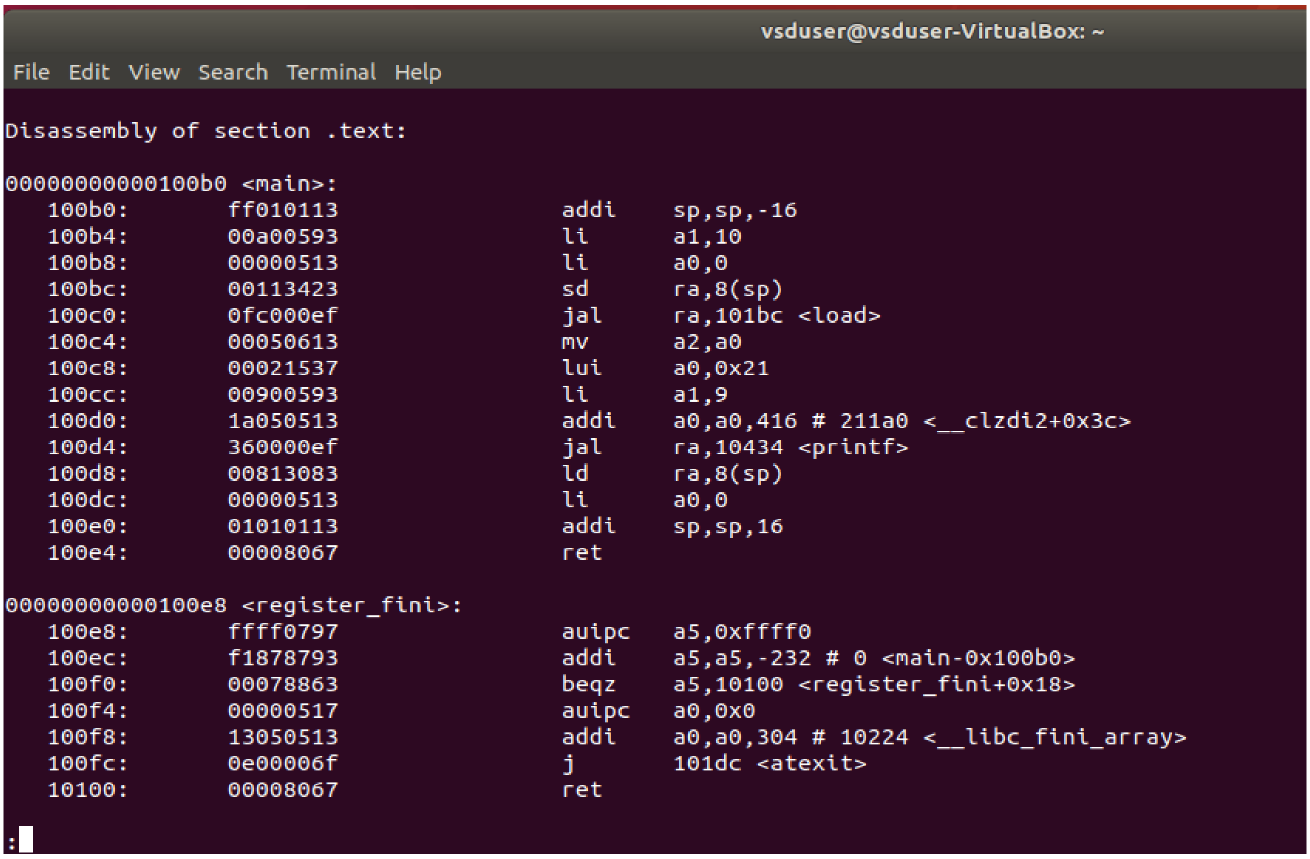
RV_D2SK2_L3_Simulate New C Program With Function Call


vsduser@vsduser-VirtualBox: ~
File Edit View Search Terminal Help

```
vsduser@vsduser-VirtualBox:~$ cat 1to9_custom.c
#include <stdio.h>

extern int load(int x, int y);

int main() {
        int result = 0;
        int count = 9;
        result = load(0x0, count+1);
        printf("Sum of numbers from 1 to %d is %d\n", count, result);
}vsduser@vsduser-VirtualBox:~$ cat load.S
.section .text
.global load
.type load, @function

load:
        add     a4, a0, zero
        add     a2, a0, a1
        add     a3, a0, zero
loop:   add     a4, a3, a4
        addi    a3, a3, 1
        blt     a3, a2, loop
        add     a0, a4, zero
        retvsduser@vsduser-VirtualBox:~$
```


vsduser@vsduser-VirtualBox: ~
File Edit View Search Terminal Help

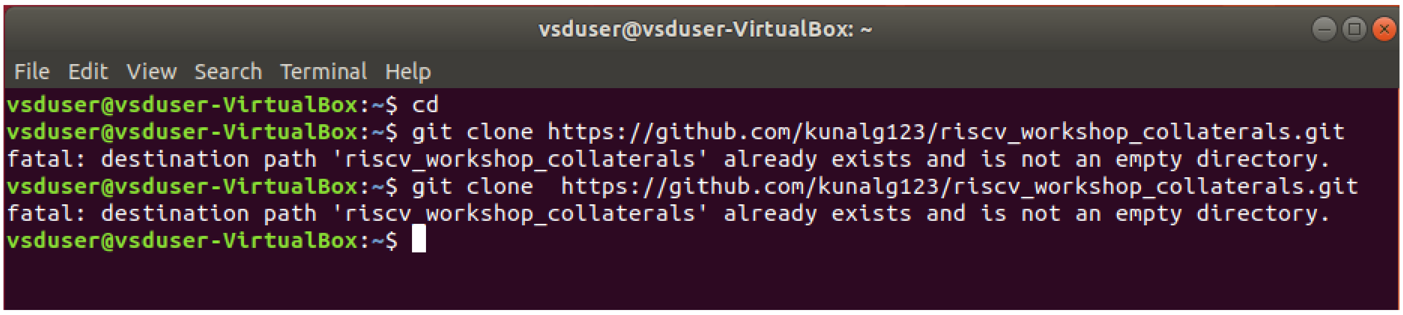
```
vsduser@vsduser-VirtualBox:~$ riscv64-unknown-elf-gcc -Ofast -mabi=lp64 -march=rv64i -o 1to9_custom.o 1to9_custom.c load.S
vsduser@vsduser-VirtualBox:~$ spike pk 1to9_custom.o
bbl loader
Sum of numbers from 1 to 9 is 45
vsduser@vsduser-VirtualBox:~$ riscv-unknown-elf-odjdump -d 1to9_custom.o | less
vsduser@vsduser-VirtualBox:~$ riscv-unknown-elf-odjdump -d 1to9_custom.o | less
vsduser@vsduser-VirtualBox:~$
vsduser@vsduser-VirtualBox:~$ riscv64-unknown-elf-odjdump -d 1to9_custom.o | less
vsduser@vsduser-VirtualBox:~$
vsduser@vsduser-VirtualBox:~$ riscv64-unknown-elf-objdump -d 1to9_custom.o | less
```


vsduser@vsduser-VirtualBox: ~
File Edit View Search Terminal Help

```
Disassembly of section .text:

00000000000100b0 <main>:
   100b0:       ff010113                addi    sp,sp,-16
   100b4:       00a00593                li      a1,10
   100b8:       00000513                li      a0,0
   100bc:       00113423                sd      ra,8(sp)
   100c0:       0fc000ef                jal     ra,101bc <load>
   100c4:       00050613                mv      a2,a0
   100c8:       00021537                lui     a0,0x21
   100cc:       00900593                li      a1,9
   100d0:       1a050513                addi    a0,a0,416 # 211a0 <__clzdi2+0x3c>
   100d4:       360000ef                jal     ra,10434 <printf>
   100d8:       00813083                ld      ra,8(sp)
   100dc:       00000513                li      a0,0
   100e0:       01010113                addi    sp,sp,16
   100e4:       00008067                ret

00000000000100e8 <register_fini>:
   100e8:       ffff0797                auipc   a5,0xffff0
   100ec:       f1878793                addi    a5,a5,-232 # 0 <main-0x100b0>
   100f0:       00078863                beqz    a5,10100 <register_fini+0x18>
   100f4:       00000517                auipc   a0,0x0
   100f8:       13050513                addi    a0,a0,304 # 10224 <__libc_fini_array>
   100fc:       0e00006f                j       101dc <atexit>
   10100:       00008067                ret

:
```

RV-D2SK3 - Basic verification flow using Verilog

RV_D2SK3_L1_Lab To Run C-Program On RISC-V CPU


vsduser@vsduser-VirtualBox: ~
File Edit View Search Terminal Help

```
vsduser@vsduser-VirtualBox:~$ cd
vsduser@vsduser-VirtualBox:~$ git clone https://github.com/kunalg123/riscv_workshop_collaterals.git
fatal: destination path 'riscv_workshop_collaterals' already exists and is not an empty directory.
vsduser@vsduser-VirtualBox:~$ git clone  https://github.com/kunalg123/riscv_workshop_collaterals.git
fatal: destination path 'riscv_workshop_collaterals' already exists and is not an empty directory.
vsduser@vsduser-VirtualBox:~$ 
```