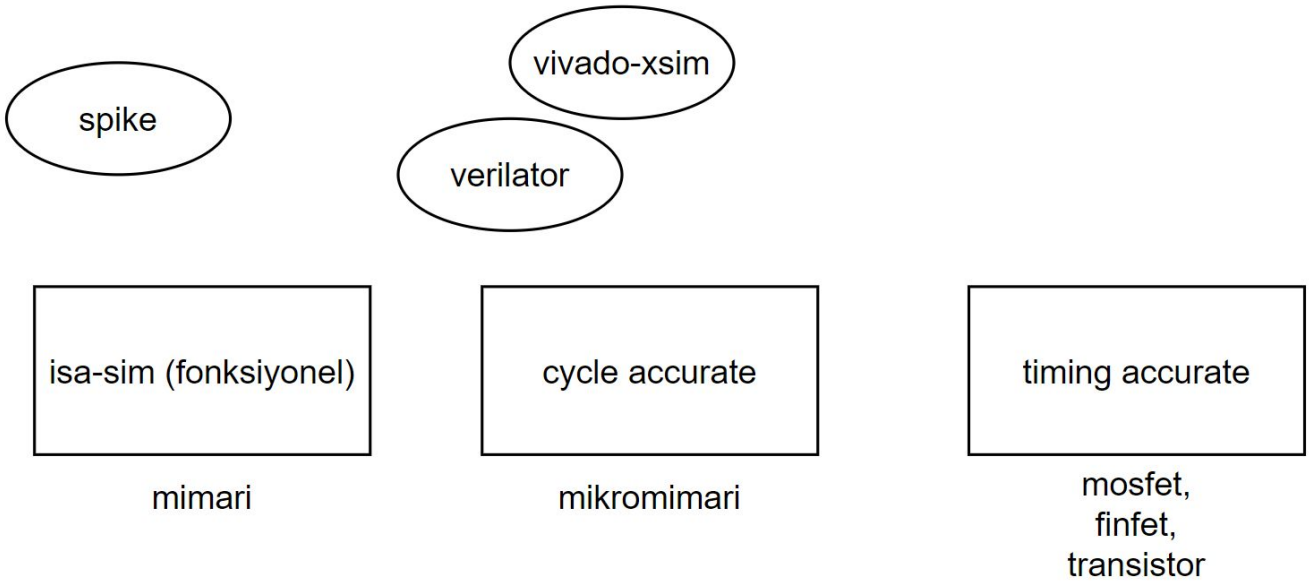
Spike Cosim

İçindekiler

Spike



Spike bir isa simülator

Cycle accurate değil

Buyrukların etkilerini görmek için bir referans model

Spike

```
> spike -d --log-commits outputs/cf-machine-code.elf
warning: tohost and fromhost symbols not in ELF; can't communicate with target
../riscv/sim.cc:581 object at:0x7ffd0f7d7010 sim.cfg.startpc.hasval: 0
sim.cc/ sim_t::set_rom()/ start pc val: 2147500032
after start() call in htif_t::run() tohost_addr: 0
(spike) r 5
core   0: 0x0000000000001000 (0x00000297) auipc   t0, 0x0
core   0: 3 0x0000000000001000 (0x00000297) x5  0x0000000000001000
core   0: 0x0000000000001004 (0x02028593) addi    a1, t0, 32
core   0: 3 0x0000000000001004 (0x02028593) x11 0x0000000000001020
core   0: 0x0000000000001008 (0xf1402573) csrr    a0, mhartid
core   0: 3 0x0000000000001008 (0xf1402573) x10 0x0000000000000000
core   0: 0x000000000000100c (0x0182b283) ld      t0, 24(t0)
core   0: 3 0x000000000000100c (0x0182b283) x5  0x0000000080004000 mem 0x0000000000
001018
core   0: 0x0000000000001010 (0x00028067) jr      t0
core   0: 3 0x0000000000001010 (0x00028067)
(spike) reg 0 mstatus
0x0000000a00000000
(spike) reg 0 sstatus
0x0000000200000000
(spike)
```

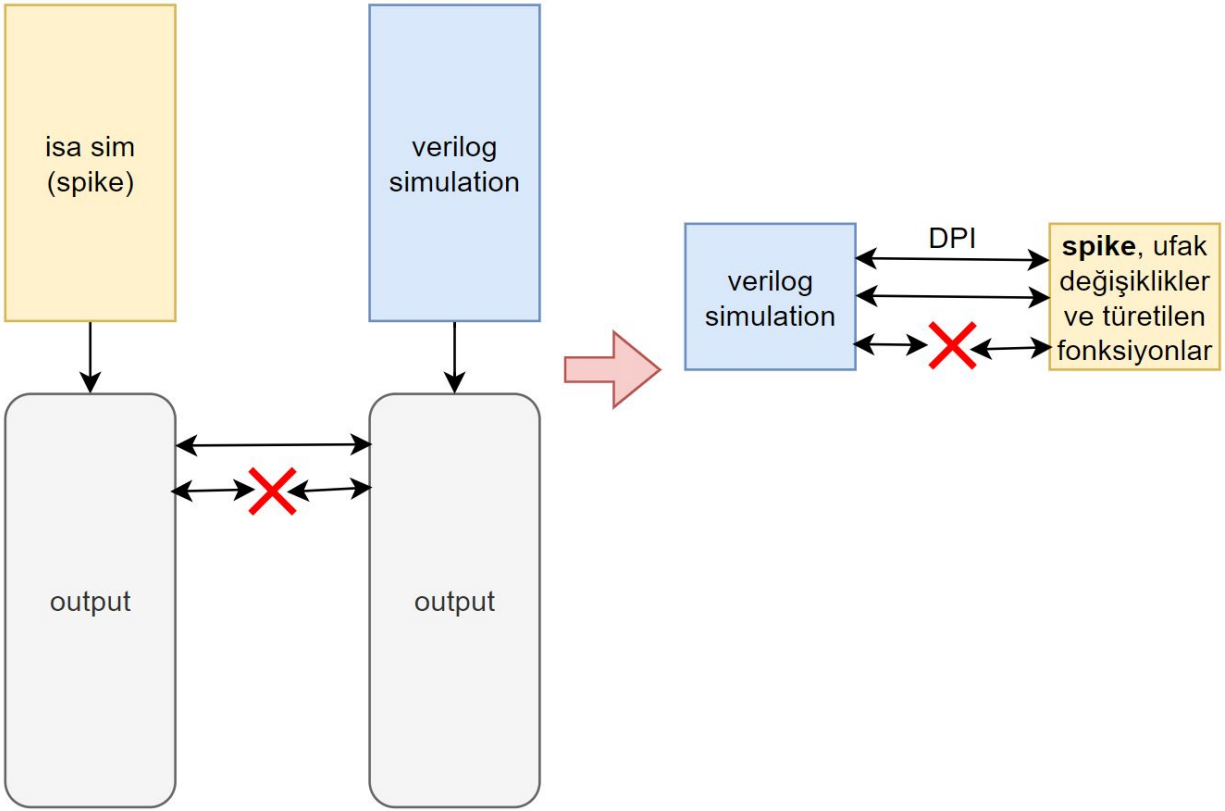
Spike bir terminal uygulaması. flag'leri ve yürüteceğimiz riscv için derlenmiş .elf dosyasını veriyoruz.

Yanda "Interactive mod" seçeneği ile kullanılıyor.

Amacımız

Spike'ın test kodunu tek parça hâlinde kendi başına çalıştırması yerine Verilog testbench'lerinden adım adım çalıştırılabilecek bir arayüzünü tasarlamak

Rtl simülasyonu ve isa simülasyonunu birleştirmek



DPI (SystemVerilog Direct Programming Interface)

c tarafında tanımlanan fonksiyonları SystemVerilog'dan nasıl "import"layacağımızı belirleyen bir arayüz, bir standart. (sv. Language standart'ta bölüm 35)

"Export" kısmı da var, cosim'de hiç kullanmadık.

Sentez (derleme/simülasyon) için kullandığımız araç, bu import'ların çalışmasını sağlıyor.

```
import "DPI-C" function void wait_key();
```

```
void wait_key()
{
  std::cout << "press any key to continue..." << std::endl;
  std::cin.get();
}
```

DPI Örnekler

```
void get_log_mem_read(svBitVecVal* log_mem_read_o,
    int* inserted_elements_o, int processor_id);
void get_log_mem_write(svBitVecVal* log_mem_write_o,
    int* inserted_elements_o, int processor_id);
void get_log_reg_write(svBitVecVal* log_reg_write_o,
    int* inserted_elements_o, int processor_id);
void get_pc(svBitVecVal* pc_o, int processor_id);
void init();
void open_array_example(const svOpenArrayHandle arr);
svBit simulation_completed();
void step();
void wait_key();
```

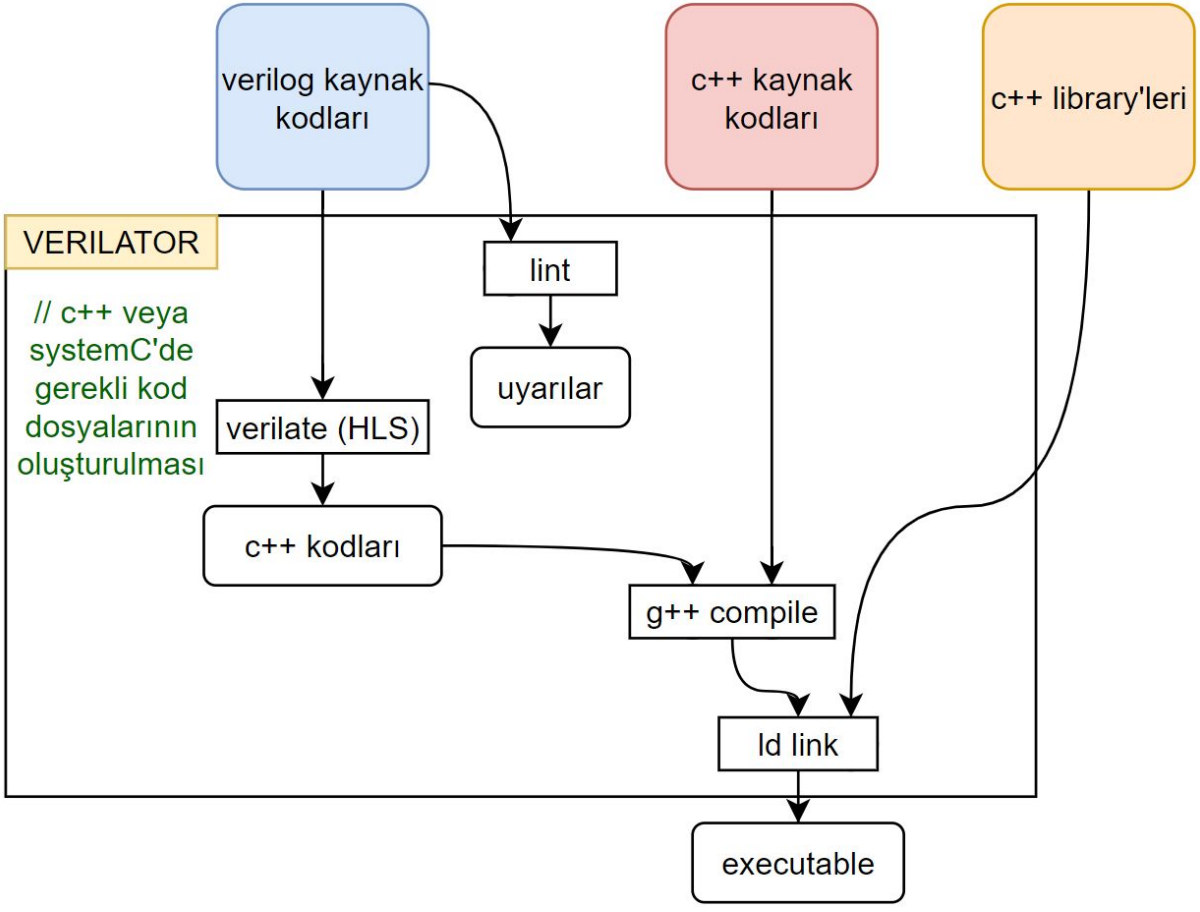
```
import "DPI-C" function void init();
import "DPI-C" function void step();
import "DPI-C" function bit simulation_completed();
import "DPI-C" function void wait_key();
import "DPI-C" function void get_log_reg_write(
  output commit_log_reg_item_t log_reg_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
import "DPI-C" function void get_log_reg_write(
  output commit_log_reg_item_t log_reg_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
import "DPI-C" function void get_log_mem_write(
  output commit_log_mem_item_t log_mem_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
import "DPI-C" function void get_pc(
  output reg_t pc_o,
  input int processor_id = 0
);
import "DPI-C" function void open_array_example(
  input bit [31:0] arr[3][]
);
```

Verilator

Verilator, verilog'dan C++'a HLS yapan ve bu HLS çıktılarını derleyip linkleme işlemini otomatize eden bir araç.

Verilator dokümantasyon, repo

Aynı zamanda harici c++ kaynak kodlarını/library'lerini de derleme ve linkleme işlemine dahil edebiliyor




```
compile_$(1):
	verilator -O2 -CFLAGS -DARGS_FILE_PATH=\\\" $(CURDIR)/log/args.txt\\\" -CFLAGS
"$(INC_DIRS)" --Mdir obj_dir_$(1) --binary +1800-2017ext+sv  $(SRC_FILES)
$(TB_DIR)/$(1).sv $(CPP_FILES) $(LIBRARIES) --top $(1) --prefix $(1)  -o $(1).exe
```

Verilator - DPI Header

Verilator, verilog kodlarında geçen “DPI import” ifadeleri için oluşturduğu DPI header’ına fonksiyon imzası (prototype) olarak ekler.

spike-cosim > cosim > obj_dir_cosim_ornek_kullanim > C cosim_ornek_kullanim__Dpi.h > ...

```
1  // Verilated -*- C++ -*-
2  // DESCRIPTION: Verilator output: Prototypes for DPI import and export functions.
3  //
4  // Verilator includes this file in all generated .cpp files that use DPI functions.
5  // Manually include this file where DPI .c import functions are declared to ensure
6  // the C functions match the expectations of the DPI imports.
7
8  #ifndef VERILATED_COSIM_ORNEK_KULLANIM__DPI_H_
9  #define VERILATED_COSIM_ORNEK_KULLANIM__DPI_H_  // guard
10
11 #include "svdpi.h"
12
13 #ifdef __cplusplus
14 extern "C" {
15 #endif
16     // DPI IMPORTS
17     // DPI import at src/pkg/cosim_pkg.sv:70:32
18     extern void wait_key();
```

Verilator - DPI header'ının include'lanması

Biz bu fonksiyonları tanımlayacağımız dosyada verilator'ün oluşturduğu DPI header'ı #include'larız

```
#include "../../obj_dir_cosim_ornek_kullanim/cosim_ornek_kullanim__Dpi.h"
```

```
void wait_key()
{
    std::cout << "press any key to continue..." << std::endl;
    std::cin.get();
}
```

Elimizde Ne Var

Spike’ın simülasyonunu SystemVerilog testbench’lerinden DPI ile kontrol edebileceğimiz bir arayüz.

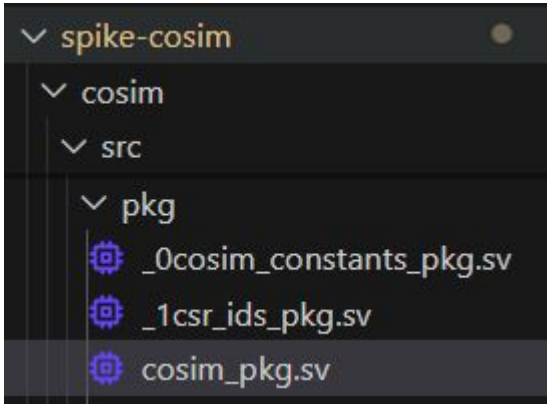
```
import "DPI-C" function void init();

import "DPI-C" function void step();

import "DPI-C" function bit simulation_completed();
```

```
import "DPI-C" function void get_log_reg_write(
  output commit_log_reg_item_t log_reg_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
```

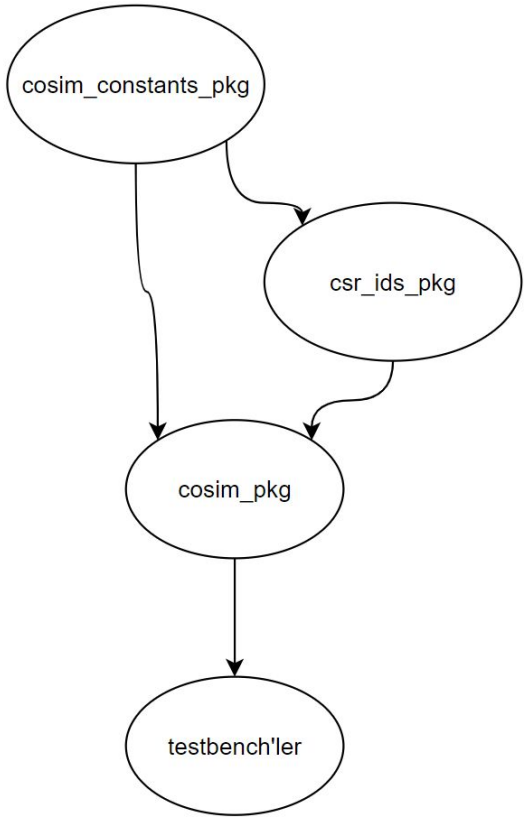
Elimizde Ne Var

```
import "DPI-C" function  void get_log_mem_read(
  output commit_log_mem_item_t log_mem_read_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);

import "DPI-C" function void get_log_mem_write(
  output commit_log_mem_item_t log_mem_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);

import "DPI-C" function void get_pc(
  output reg_t pc_o,
  input int processor_id = 0
);
```

Cosim Verilog Tarafı



_0, _1 gibi isimlendirme, verilator'e dosyaları hiyerarşik sıraya göre verebilmek için



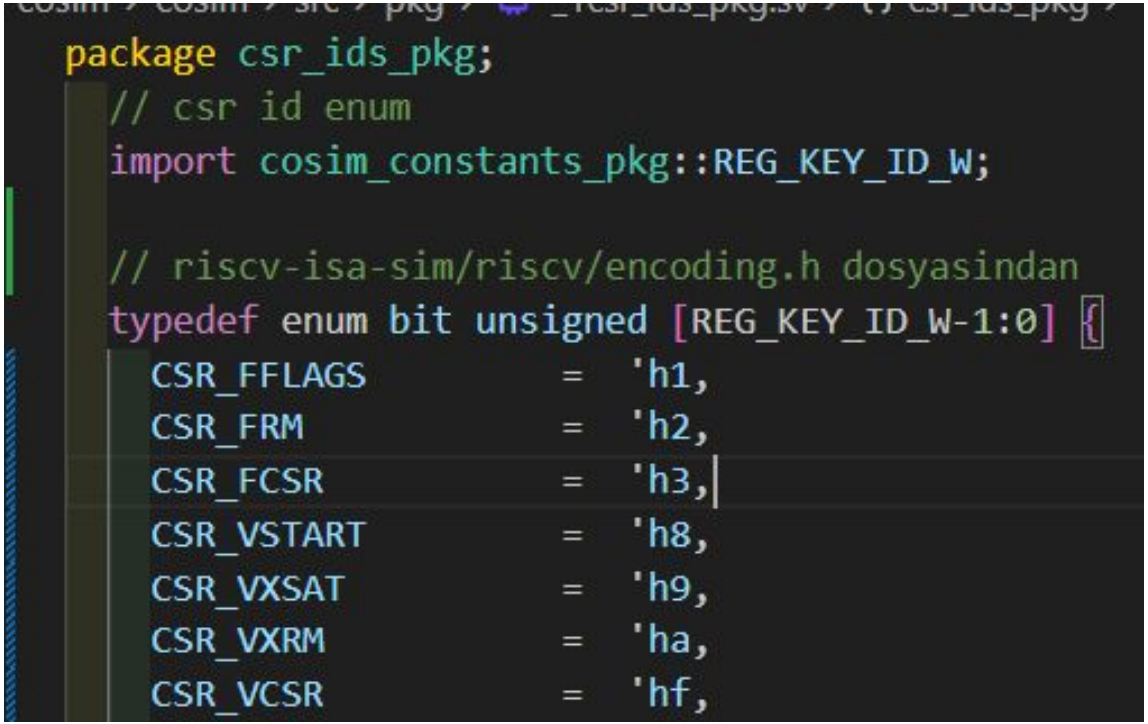
Cosim_pkg'yi import'layan testbench'te isim kalabalığı olmaması için cosim_constants_pkg ayrı dosyada.

csr_ids_pkg çok uzun olduğu için ayrı dosyada.

```
compile_$(1):
  verilator -O2 -CFLAGS -DARGS_FILE_PATH=\\\" $(CURDIR)/log/args.txt\\\" -CFLAGS
"$(INC_DIRS)" --Mdir obj_dir_$(1) --binary +1800-2017ext+sv  $(SRC_FILES)
$(TB_DIR)/$(1).sv $(CPP_FILES) $(LIBRARIES) --top $(1) --prefix $(1)  -o $(1).exe
```

csr_ids_pkg

449 tane



```
package csr_ids_pkg;
  // csr id enum
  import cosim_constants_pkg::REG_KEY_ID_W;

  // riscv-isa-sim/riscv/encoding.h dosyasindan
  typedef enum bit unsigned [REG_KEY_ID_W-1:0] {
    CSR_FFLAGS          =  'h1,
    CSR_FRM             =  'h2,
    CSR_FCSR            =  'h3,
    CSR_VSTART          =  'h8,
    CSR_VXSAT           =  'h9,
    CSR_VXRM            =  'ha,
    CSR_VCSR            =  'hf,
```

cosim_pkg Fonksiyonları

```
// args.txt dosyasindan okunan command line argumanlarla
// spike simulation olusturur. command line argumanlarin
// hangi dosyadan okundugunu
// cosim/src/cpp/cosimif.cc }} init()
// fonksiyonunun tanimindan degistirebilirsiniz.
import "DPI-C" function void init();
```

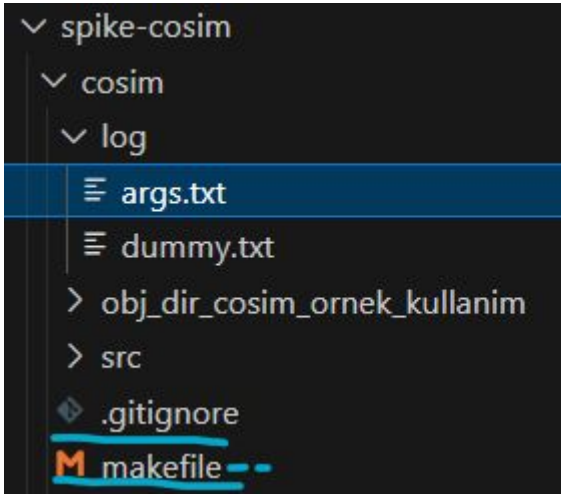
```
22  void init()
23    {
24      #ifndef ARGS_FILE_PATH
25      #define ARGS_FILE_PATH "args.txt"
26      #warning ARGS_FILE_PATH is not defined. Using default value: "args.txt"
27      #endif
28
```

spike-cosim/ cosim/src/ cpp/cosimif.cc

spike-cosim > cosim > log > args.txt

```
1   spike /home/usr1/spike-cosim/ornek_test_girdileri/pk_olmadan/outputs/hello.elf
2   spike pk /home/usr1/spike-cosim/ornek_test_girdileri/fromhost_tohost_test/a.out
3
```

args.txt Dosyası



spike-cosim/cosim/makefile

```
verilator -O2 -CFLAGS -DARGS_FILE_PATH=\\\"$(CURDIR)/log/args.txt\\\"
```

spike-cosim/cosim/src/cpp/cosimif.cc

```
22  void init()
23  {
24      #ifndef ARGS_FILE_PATH
25      #define ARGS_FILE_PATH "args.txt"
26      #warning ARGS_FILE_PATH is not defined. Using default value: "args.txt"
27      #endif
28
```

cosim_pkg Fonksiyonları

```
// spike simulation'u bir adim ilerletir.
import "DPI-C" function void step();
```

```
// simulation'da kosan kod, exitcode gonderdiyse 1 dondurur.
import "DPI-C" function bit simulation_completed();
```

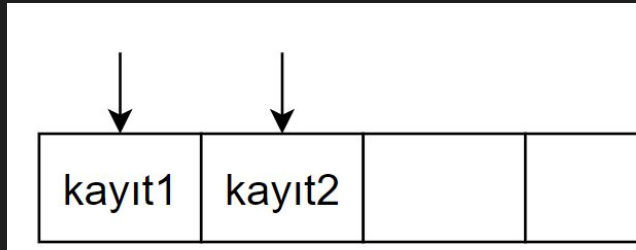
```
// cosim'in bir parcasi degil, adim adim ilerletip incelemek icin koydum.
// testbench'imde kullaniyorum. "devam etmek icin bir tusa basiniz" yapmaya yariyor.
import "DPI-C" function void wait_key();
```

```
import "DPI-C" function void get_pc(
  output reg_t pc_o,
  input int processor_id = 0
);
```

cosim_pkg Fonksiyonları

```
// son simulation adiminda yapilan register write kayitlarini output parametresine yazar
// kac tane eleman eklendiyse sayisini inserted_elements_o'ya yazar.
import "DPI-C" function void get_log_reg_write(
  output commit_log_reg_item_t log_reg_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
import "DPI-C" function  void get_log_mem_read(
  output commit_log_mem_item_t log_mem_read_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
// son yapilan step'teki memory write islemleri.
import "DPI-C" function void get_log_mem_write(
  output commit_log_mem_item_t log_mem_write_o[CommitLogEntries],
  output int inserted_elements_o,
  input int processor_id = 0
);
```

16

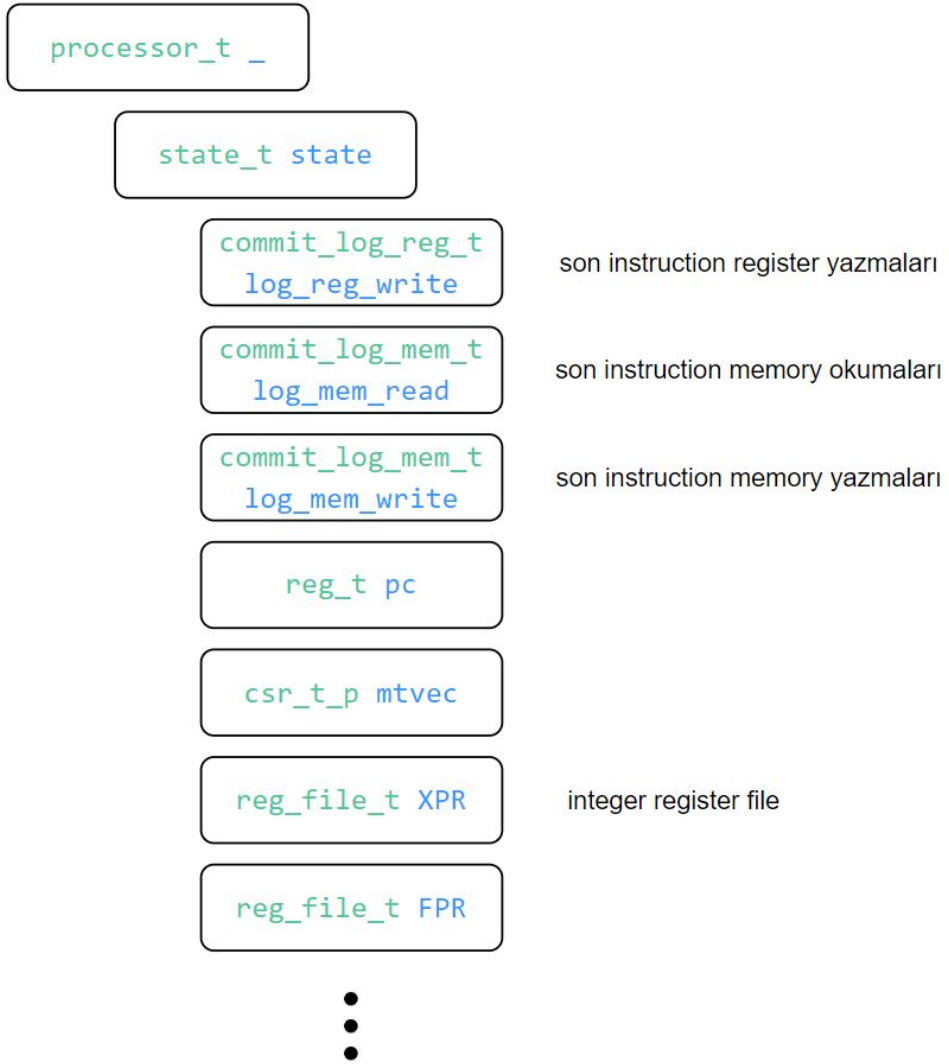


inserted_elements_o <-- 2

cosim_pkg Türler

Spike tarafında yapılan işlem kayıtları, processor'ün state'i altında bu commit_log_reg_t ve commit_log_mem_t türleri ile tutuluyor.

Cosim tarafında bunların muadilleri tanımlanıyor.



cosim_pkg Türler

```
// addr, value, size
typedef std::vector<std::tuple<reg_t, uint64_t, uint8_t>> commit_log_mem_t;
```

```
typedef struct packed {
  reg_t addr;
  reg_t wdata;
  bit [55:0] reserved; // c tarafindaki alignment'a uydurmak icin
  byte unsigned len;
} commit_log_mem_item_t;
```

Cosim_pkg Türler

```
// regnum, data
typedef std::unordered_map<reg_t, freg_t> commit_log_reg_t;
```

```
typedef bit unsigned [FREG_W-1:0] freg_t;
```

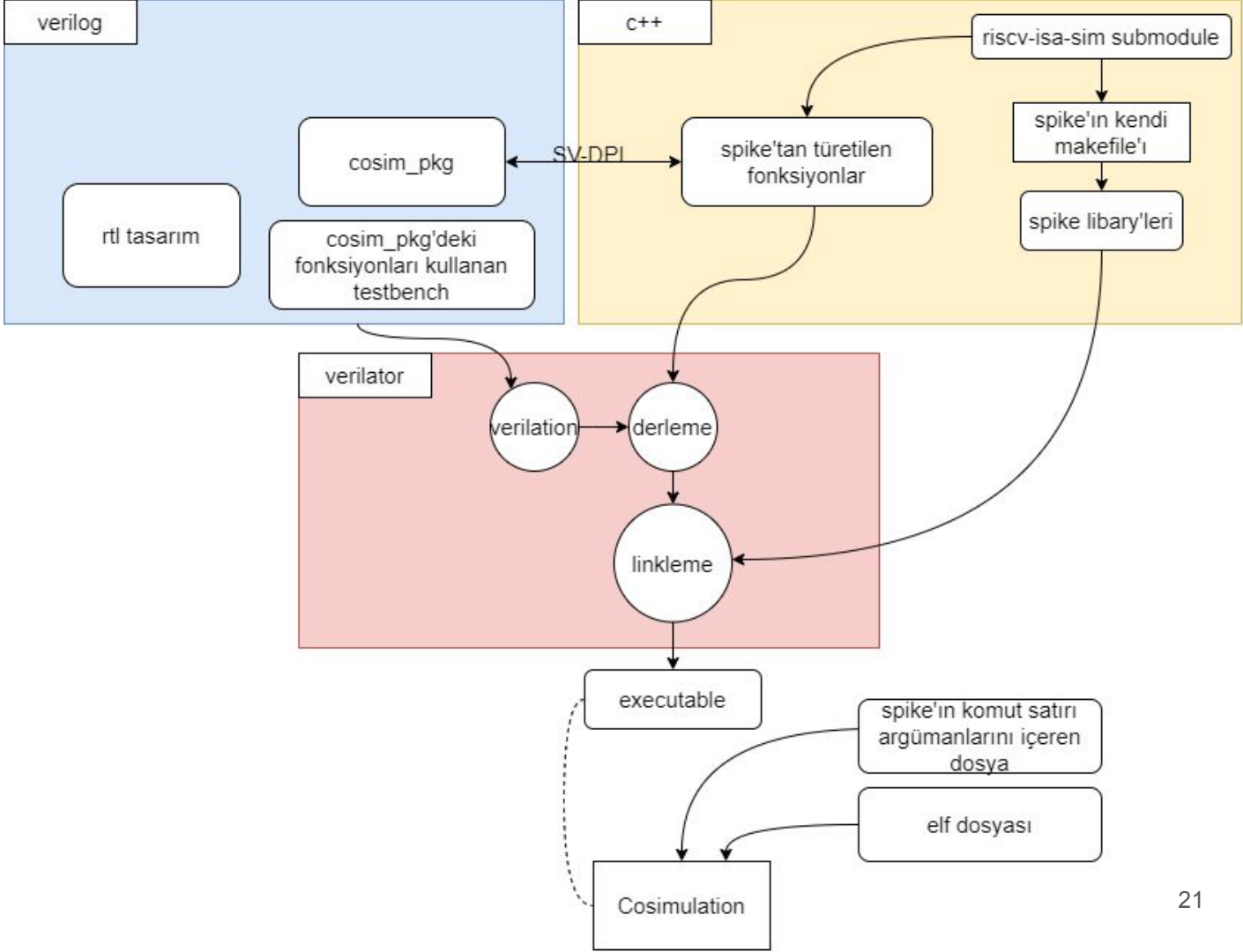
```
// riscv-isa-sim/riscv/decode_macros.h
typedef enum bit unsigned [REG_KEY_TYPE_W-1:0] {
  XREG      = REG_KEY_TYPE_W'('b0000),
  FREG      = REG_KEY_TYPE_W'('b0001),
  VREG      = REG_KEY_TYPE_W'('b0010),
  VREG_HINT = REG_KEY_TYPE_W'('b0011),
  CSR       = REG_KEY_TYPE_W'('b0100)
} reg_key_type_e;
```

```
typedef struct packed {
  reg_id_t reg_id;
  reg_key_type_e reg_type;
} reg_key_t;
```

```
// csr'larin ozel id'leri var
typedef csr_ids_pkg::csr_id_e csr_id_e;
// bunu disariya gostermek istiyorum.

// diger id'ler duz 0'dan 31'e.
typedef union packed {
  bit unsigned [REG_KEY_ID_W-1:0] xr_fr_vr_id;
  csr_id_e csr_id;
} reg_id_t;
```

```
typedef struct packed {
  reg_key_t key;
  freg_t value;
} commit_log_reg_item_t;
```

Cosim Kullanımı



Verilator ile Örnek Testbench'i Derleme

```
usr1@LENOVO:~/spike-cosim/cosim
> verilator -O2 -CFLAGS -DARGS_FILE_PATH=\\\"/home/usr1/spike-cosim/cosim/log/args.txt\\\" -
CFLAGS "-I/home/usr1/spike-cosim/riscv-isa-sim/build  -I/home/usr1/spike-cosim/riscv-isa-sim
/riscv -I/home/usr1/spike-cosim/riscv-isa-sim/fesvr -I/home/usr1/spike-cosim/riscv-isa-sim/
-I/home/usr1/spike-cosim/riscv-isa-sim/softfloat -I/home/usr1/spike-cosim/riscv-isa-sim/fdt"
--Mdir obj_dir_tb_spike_link --binary +1800-2017ext+sv src/pkg/_0cosim_constants_pkg.sv src
/pkg/_1csr_ids_pkg.sv [redacted] src/pkg/cosim_pkg.sv src/tb/tb_s
pike_link.sv src/cpp/args_reader.cc src/cpp/cosim_create_sim.cc src/cpp/cosim_htif.cc src/cp
p/cosim_sim.cc src/cpp/cosimif.cc /home/usr1/spike-cosim/riscv-isa-sim/build/libspike_dasm.a
 /home/usr1/spike-cosim/riscv-isa-sim/build/libriscv.so --top tb_spike_link --prefix tb_spik
e_link -o tb_spike_link.exe
```

Verilator tarafından oluşturulan obj_dir_cosim_ornek_kullanim/cosim_ornek_kullanim.mk dosyasında

```
VM_USER_CFLAGS = \
  -DARGS_FILE_PATH=\"/home/usr1/spike-cosim/cosim/log/args.txt\" \
  -I/home/usr1/spike-cosim/riscv-isa-sim/build  -I/home/usr1/spike-cosim/
```

Testbench’te Kullanım

```
module cosim_ornek_kullanim;
  import cosim_pkg::*;

  commit_log_reg_item_t log_reg_write_from_c [CommitLogEntries];
  commit_log_mem_item_t log_mem_read_from_c [CommitLogEntries];
  commit_log_mem_item_t log_mem_write_from_c [CommitLogEntries];
  int num_elements_inserted_from_c_side; // 3'u icin de kullaniliyor.
  reg_t temp_key;
  freg_t temp_value;
  reg_t temp_pc;
```

```
  initial begin: cosimulation
    init();

    for (;;) begin: simulation_loop···

    end: simulation_loop
    $finish;
  end: cosimulation
```

Testbench’te Kullanım

Testbench’i ve diğer gerekli dosyaları verilator ile derleyip çalıştırınca init fonksiyonunun sonuna kadarki kısmın çıktıları aşağıdaki gibi

```
../src/cpp/cosimif.cc:33: reading args from file: /home/usr1/spike-cosim/cosim/log/
args.txt

--running cosim with arguments (the first line of the file):--
 spike /home/usr1/spike-cosim/ornek_test_girdileri/pk_olmadan/outputs/hello.elf
---------------------------------------
sim.cc/ sim_t::set_rom()/ start pc val: 2147483784
communication_available() is true
```

Testbench’te Kullanım

```
for (;;) begin: simulation_loop
  if (simulation_completed()) begin // htif_t::exitcode != 0
    $display("simulation completed");
    break;
  end

  get_pc(temp_pc);
  $display("pc before execution: %0h", temp_pc);

  step();

/* spike tarafinda processorlerin yaptigi islemleri incele…
*/
  wait_key();

end: simulation_loop
```

Testbench’te Kullanım

```
get_log_reg_write(log_reg_write_from_c,  num_elements_inserted_from_c_side);

for (int ii = 0; ii < num_elements_inserted_from_c_side; ii = ii + 1) begin: log_reg_write_itr

  $display("log_reg_write_from_c[%0d] reg_type: %0s",
  ii, log_reg_write_from_c[ii].key.reg_type.name);

  if (log_reg_write_from_c[ii].key.reg_type == CSR) begin
    $display("log_reg_write_from_c[%0d] csr name: %0s",
    ii, log_reg_write_from_c[ii].key.reg_id.csr_id.name);
  end else begin
    $display("log_reg_write_from_c[%0d] reg_id: %0d",
    ii, log_reg_write_from_c[ii].key.reg_id);
  end

  $display("log_reg_write_from_c[%0d].value: %0h", ii, log_reg_write_from_c[ii].value);
end
```

Testbench’te Kullanım

register işlemlerine dair testbench tarafından basılan çıktılardan biri (illegal instruction içeren bir test girdisi için):

```
pc before execution: 8000008c
log_reg_write_from_c[0] reg_type: CSR
log_reg_write_from_c[0] csr name: CSR_MTINST
log_reg_write_from_c[0].value: 0
log_reg_write_from_c[1] reg_type: CSR
log_reg_write_from_c[1] csr name: CSR_MTVAL2
log_reg_write_from_c[1].value: 0
log_reg_write_from_c[2] reg_type: CSR
log_reg_write_from_c[2] csr name: CSR_MTVAL
log_reg_write_from_c[2].value: c20022f3
log_reg_write_from_c[3] reg_type: CSR
log_reg_write_from_c[3] csr name: CSR_MCAUSE
log_reg_write_from_c[3].value: 2
log_reg_write_from_c[4] reg_type: CSR
log_reg_write_from_c[4] csr name: CSR_MSTATUS
log_reg_write_from_c[4].value: a00001800
log_reg_write_from_c[5] reg_type: CSR
log_reg_write_from_c[5] csr name: CSR_MEPC
log_reg_write_from_c[5].value: 8000008c
press any key to continue...
```

Bir diğeri

```
pc before execution: 1008
log_reg_write_from_c[0] reg_type: XREG
log_reg_write_from_c[0] reg_id: 10
log_reg_write_from_c[0].value: 0
press any key to continue...
```

Testbench’te Kullanım

```
get_log_mem_read(log_mem_read_from_c, num_elements_inserted_from_c_side);

for (int ii = 0; ii < num_elements_inserted_from_c_side; ii = ii + 1) begin: log_mem_read_itr
  $display("log_mem_read_from_c[%0d].addr: %0h", ii, log_mem_read_from_c[ii].addr);
  $display("log_mem_read_from_c[%0d].wdata: %0h", ii, log_mem_read_from_c[ii].wdata);
  $display("log_mem_read_from_c[%0d].len: %0d", ii, log_mem_read_from_c[ii].len);
end

get_log_mem_write(log_mem_write_from_c, num_elements_inserted_from_c_side);

for (int ii = 0; ii < num_elements_inserted_from_c_side; ii = ii + 1) begin: log_mem_write_itr
  $display("log_mem_write_from_c[%0d].addr: %0h", ii, log_mem_write_from_c[ii].addr);
  $display("log_mem_write_from_c[%0d].wdata: %0h", ii, log_mem_write_from_c[ii].wdata);
  $display("log_mem_write_from_c[%0d].len: %0h", ii, log_mem_write_from_c[ii].len);
```

RISCV Proxy-Kernel

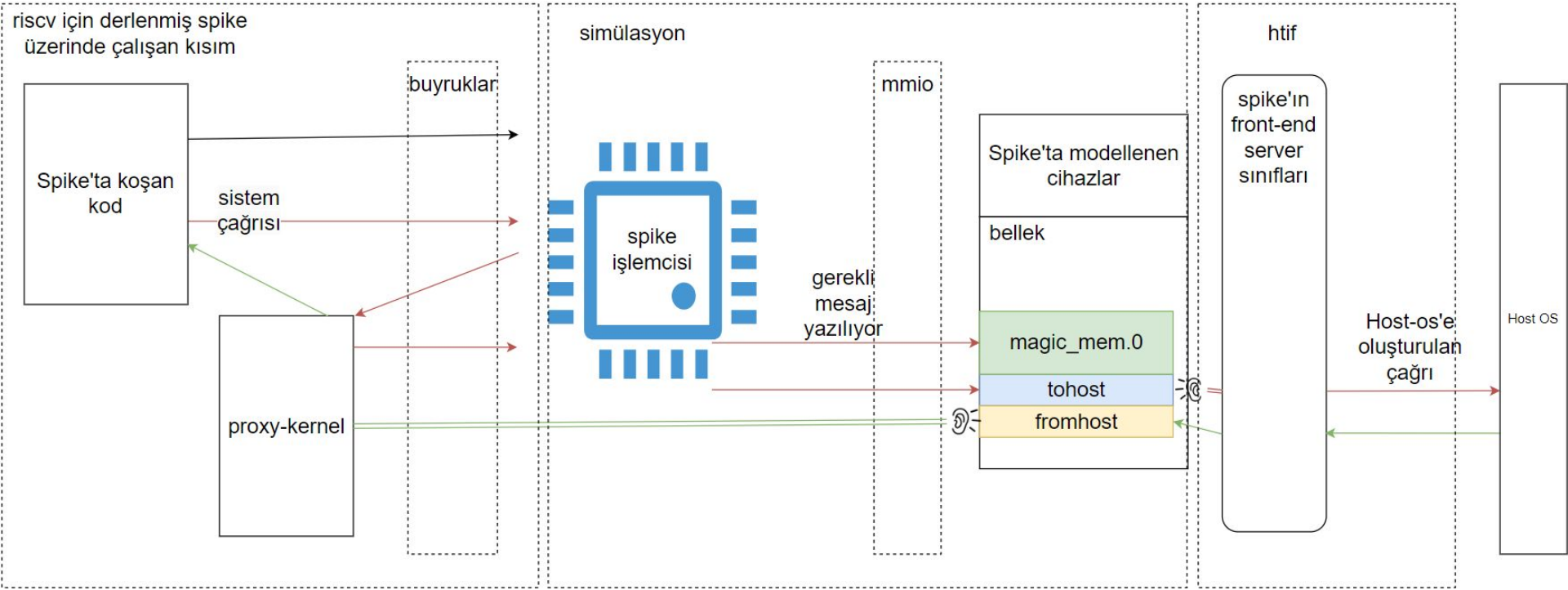
Spike’ta sadece bazı temel cihazlar modellenir. Processor, plic, clint, ns16550, memory gibi.

Bunlar sistem çağrılarını desteklemeye yeterli değiller.

O yüzden spike’ta sistem çağrısı yapan bir kod koşmak istediğimizde bunları host-os’e yönlendirecek bir mekanizmaya ihtiyaç var.

Bu işi proxy-kernel ve spike’ın frontend server kütüphanesi hallediyor.

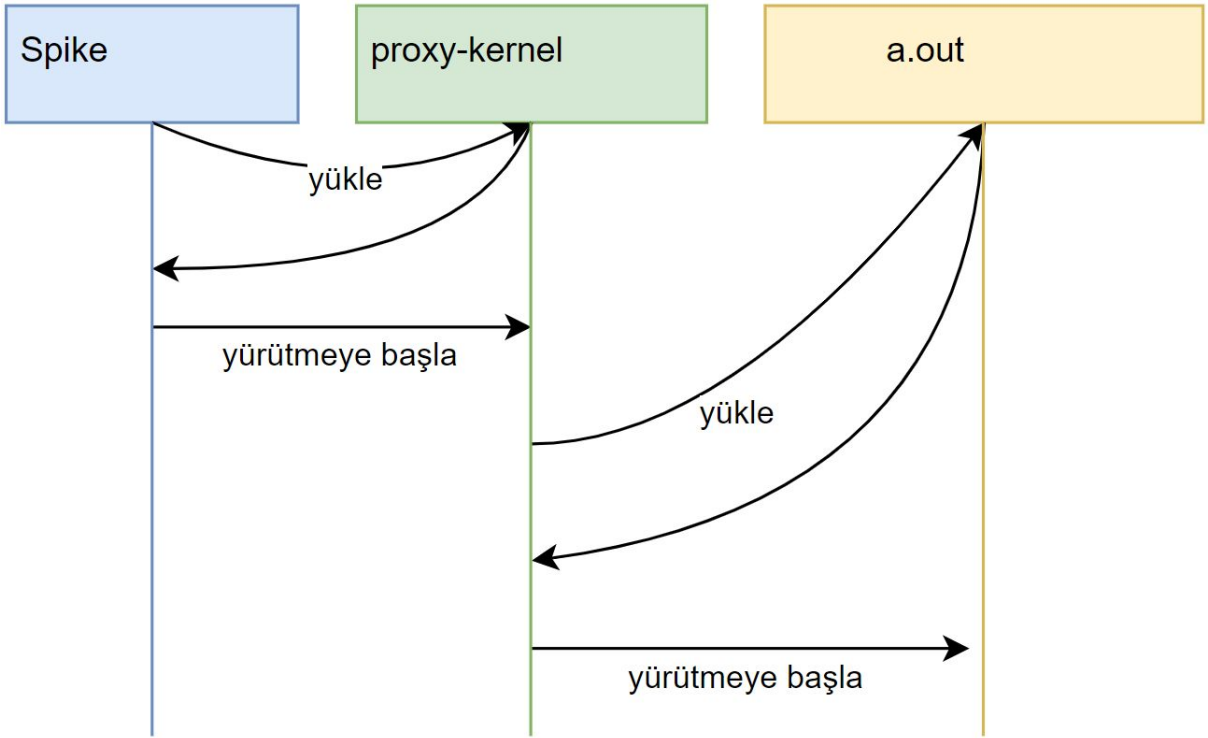
RISC V Proxy-Kernel



https://thenounproject.com/browse/icons/term/listen/

RISCV Proxy-Kernel

```
spike pk /home/usr1/spike-cosim/ornek_test_girdileri/fromhost_tohost_test/a.out
```



Soldaki komutu çalıştırdığımızda spike önce proxy-kernel'i process'in bellek alanına yüklüyor, sonra kontrolü pk'ye devrediyor. Sonra pk a.out'u yükleyip kontrolü ona devrediyor.

Baremetal Kodu Sonlandırma

bit cosim_pkg::simulation_completed fonksiyonu spike'ta koşan kodun exit sistem çağrısı yapıp yapmadığını host-target interface'in exitcode alanı üzerinden dinler. Normalde exit sistem çağrısı pk tarafından yapılır. Biz baremetal kodda pk'in yaptığı işi yapmak için şunları ekleriz: (aksi hâlde simulation_completed fonksiyonu işlevsizdir)

```
volatile static long long int magic_mem[8]; // bunun ismi onemli degil.
volatile static long long int* tohost __attribute__((used));// bunlarin
volatile static long long int* fromhost __attribute__((used));// onemli
```

Baremetal Kodu Sonlandırma

Ve programı sonlandırmak istediğimiz yerde şu işlemleri yapan fonksiyonu çağırırız:

```
void baremetal_exit(long long int exit_code){
  magic_mem[1] = exit_code;
  magic_mem[0] = 93; // 93: exit
  // see riscv-isa-sim/fesvr/syscall.cc }}
  // syscall_t::syscall_t
  for (int i = 2; i < 8; i++)
    magic_mem[i] = 0;
  tohost = magic_mem;
  while (!fromhost);
  fromhost = 0;
  while (1);
}
```