

sdfirm:

RISC-V验证方法论与实践

郑律 ‘ZETALOG’

软件历史

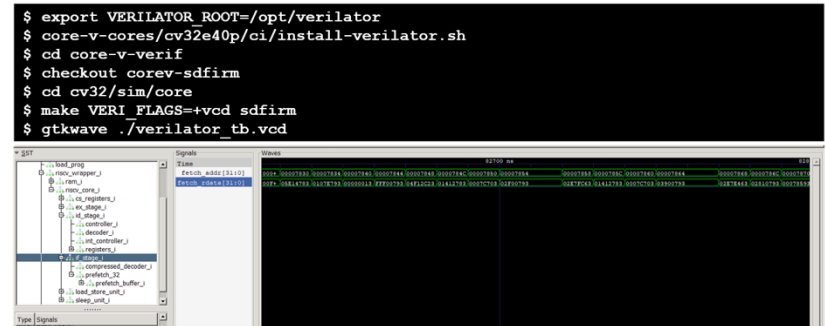
- 2011年开源USB外设固件：
 - 支持SDCC/GCC
 - AT89C5122: 128B IRAM, 256B ERAM
 - ISO7816-3协议栈/Secure CCID读卡器
 - MSD U盘/HID键盘
 - <https://sdfirm.sourceforge.net/>
- 2019年移植RISC-V系统验证框架/教学系统
 - 启动系统
 - ZSBL/FSBL: 用作ROM, 启动固件
 - BBL: SBI实现
 - IC测试盲点: 中断/异步裸金属系统验证
 - 增强的RIS-V系统验证功能
 - 简单benchmarking
 - RISC-V ISA compliance
 - RISC-V litmus
 - GEM切片
 - <https://github.com/zetalog/sdfirm>



RI5CY的中断控制器

- RI5CY (CV32E40P) 支持Vectored IRQ, 不适合执行一些通用测试
- 为RI5CY增加通用中断支持
 - <https://github.com/openhwgroup/cv32e40p/pull/299>
- 为core-v-verif增加sdfirm验证支持
 - 需要安装verilator和gtkwave开发
 - <https://github.com/zetalog/core-v-verif>
 - 增加外设驱动
 - cvprint: 0x10000000
 - cvtimer: 0x15000000
 - cvtests: 0x20000000
 - <https://github.com/openhwgroup/core-v-verif/pull/166>

```
$ export VERILATOR_ROOT=/opt/verilator
$ core-v-cores/cv32e40p/ci/install-verilator.sh
$ cd core-v-verif
$ checkout corev-sdfirm
$ cd cv32/sim/core
$ make VERI_FLAGS=-vcd sdfirm
$ gtwave ./verilator tb.vcd
```



```
$ make sdfirm
./testbench_verilator \
  +*firmware=/home/zetalog/workspace/openhwgroup/core-v-verif/cv32/sim/core/../../../../cv32/tests/core/sdfirm/sdfirm.hex" \
  | tee cobj_dir/logs/sdfirm.log
scopesDump:
SCOPE 0x7ffff8194318: TOP.tb_top_verilator.riscv_wrapper.i_ram_i
SCOPE 0x7ffff8194340: TOP.tb_top_verilator.riscv_wrapper.i_ram_i.dp_ram_i
DPI-EXPORT 0x7fbfb74226c0: read_byte
DPI-EXPORT 0x7fbfb74226e0: write_byte
SCOPE 0x7ffff8194368: TOP.tb_top_verilator.riscv_wrapper.i_ram_i.read_mux

[CORE] Core settings: PULP_SECURE = 0, N_PMP_ENTRIES = 16, N_PMP_CFG 4
finished dumping memory
OpenHW Group - CV32E40P Verification
4.4.0-18362-Microsoft - 1.0.0.0

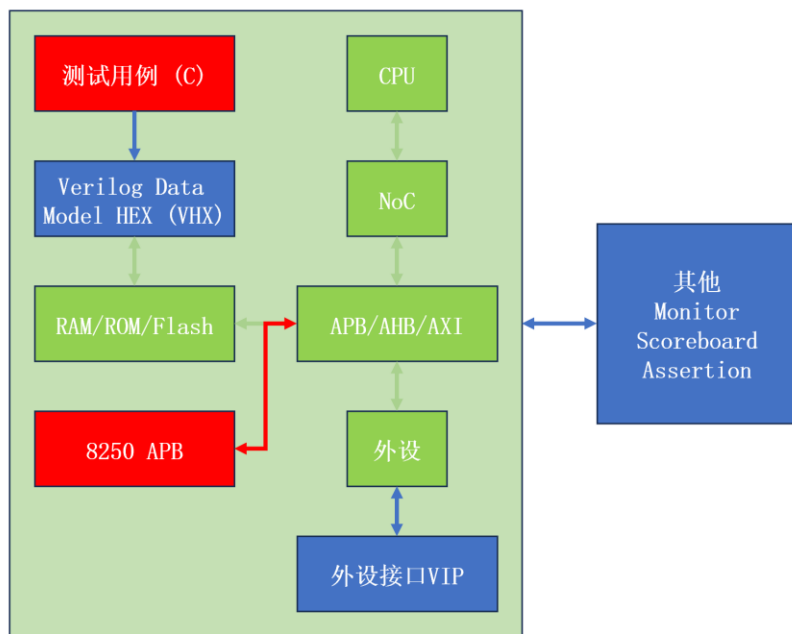
SDFIRM

EXIT SUCCESS
~/home/zetalog/workspace/openhwgroup/core-v-verif/cv32/sim/core/../../../../cv32/tb/core/tb_top_verilator.sv:83: Verilog $finish
```

软件思维验证

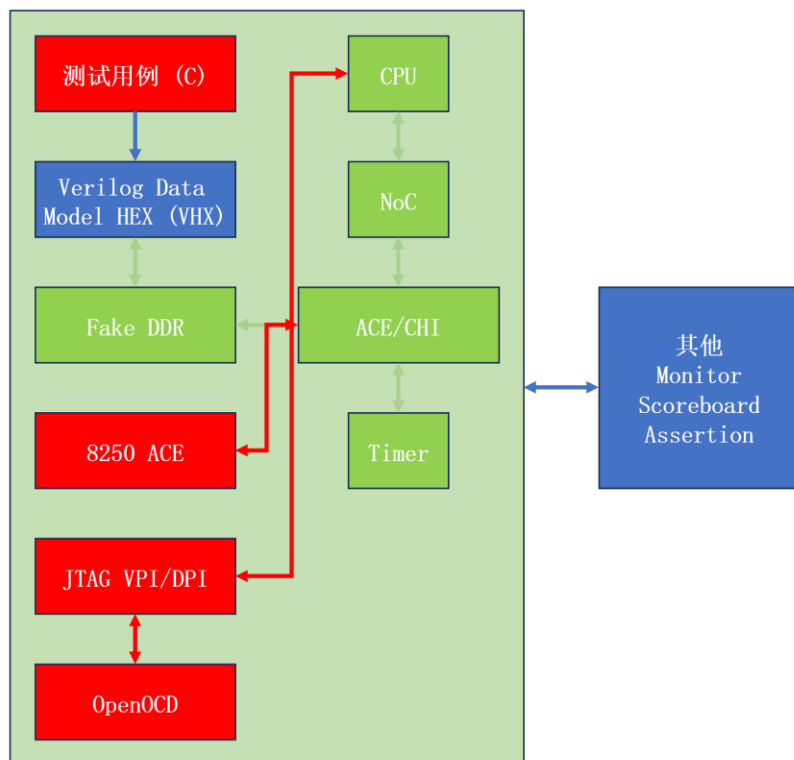
SoCDV/CPUDV test bench

SoC验证方法论



- 对硬件验证的增强：
 - 使用C测试用例，一次编写，各种验证环境执行
 - 串口修改成可读可写，可以实现异步乱序激励，更接近软件调试板卡环境
 - 外设接口VIP可以替换为产品用的外设的行为模型，更接近硅后产品形态
- 自带的测试属性：
 - 真实加载流程测出总线编址，内存bank编址错误
 - 真实时钟/复位流程测出glitch free设计问题
 - 真实多核启动流程测出Atomic设计需求缺失问题
 - 真实输入测出管脚配置设计问题引起的状态死锁
 - 真实中断处理流程测试中断设计饿死无法应答问题
- 测试用例开发支持：
 - 提供同步/异步API
 - 提供中断API
 - 提供总线配置API
 - 提供不同仿真环境下的时钟/复位的旁路和加速
 - 不同仿真环境下输入/输出频率可编程
 - 适配生态Linux/u-boot驱动
- 测试形态变化：
 - SoC测试用例的乱序执行
 - 社区软件皆可为测试激励

CPU验证方法论



- 对硬件验证的增强：
 - 使用C测试用例，一次编写，各种验证环境执行
 - 串口修改成可读可写，可以实现异步乱序激励，更接近软件调板子环境
 - 增加系统Timer用于测试操作系统的分时调度功能
 - 增加JTAG仿真硬件用于测试程序的调试功能
- 自带的测试属性：
 - 调试/跟踪规范兼容测试
 - 栈/返回地址栈的测试
 - 协程的测试
 - 系统调用的测试
 - 中断/进程上下文切换测试
 - 多任务/多虚拟机切换测试
 - Spectre实现Meltdown攻击测试
 - 缓存一致性/程序顺序/全局顺序
- 测试用例开发支持：
 - 提供缓存冲刷API
 - 提供内存屏障API
 - 提供多核元语API
 - 提供多进程/多虚拟机API
- 测试形态变化：
 - CPU测试用例的乱序执行
 - 社区软件皆可为测试激励
 - 多核一致性形式化验证

硬件思维验证

IPDV test bench

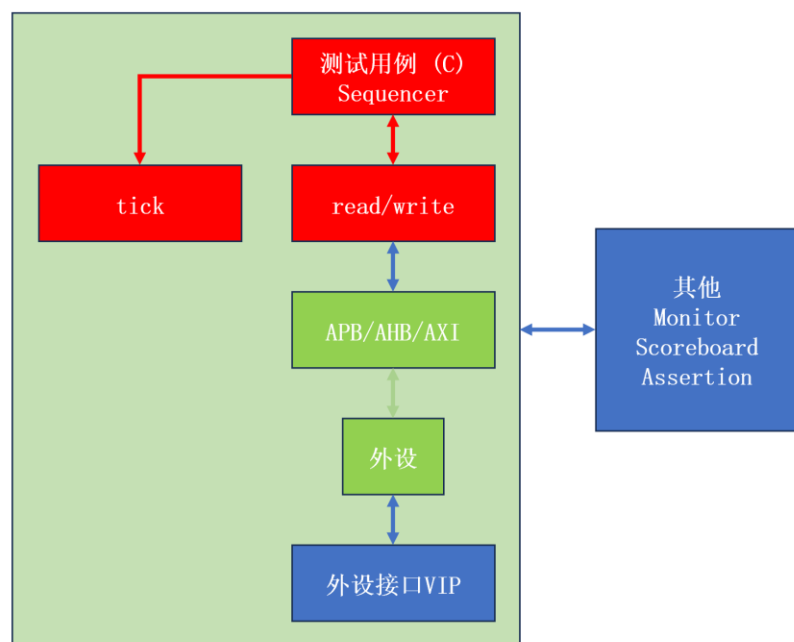
硬件验证与软件驱动结合

- DPI-C
 - IEEE 1800 Annex F: 将Foreign Language (C) 代码接入SystemVerilog
 - `#include <svdpi.h>`
 - `import (C->SV) /export (SV->C)`
 - `input/output`
 - 多使用`bit`而非`logic`
- BFM (Bus Function Model) 激励模型
 - 驱动对寄存器的访问以BFM激励实现
 - 驱动对内存的访问以MemoryModel实现

SystemVerilog type	C type
<code>byte</code>	<code>char</code>
<code>shortint</code>	<code>short int</code>
<code>int</code>	<code>int</code>
<code>longint</code>	<code>long long</code>
<code>real</code>	<code>double</code>
<code>shortreal</code>	<code>float</code>
<code>chandle</code>	<code>void*</code>
<code>string</code>	<code>const char*</code>
<code>bit^a</code>	<code>unsigned char</code>
<code>logic^a/reg</code>	<code>unsigned char</code>

^aEncodings for `bit` and `logic` are given in file `svdpi.h`. Reg parameters can use the same encodings as logic parameters.

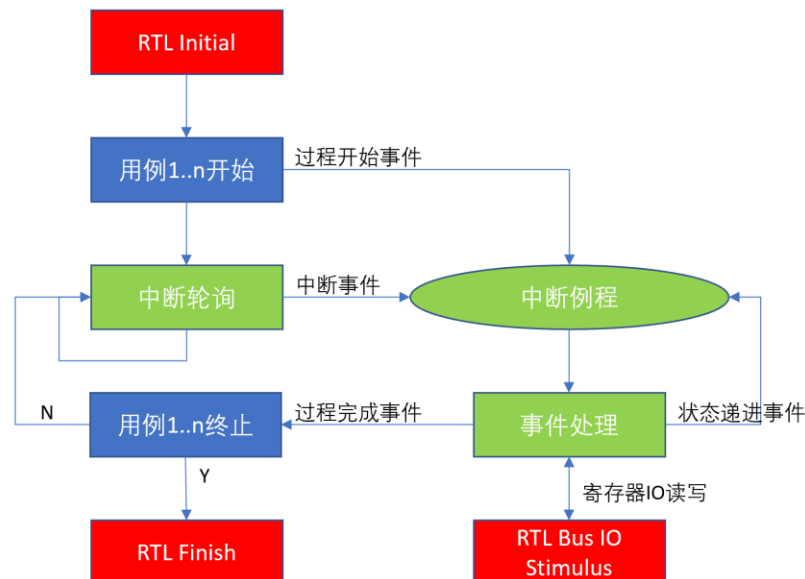
IP验证方法论



- 对硬件验证的增强：
 - 增加了IO读写sequence实现BFM
 - 利用读写API的地址基址访问不同的硬件总线接口
 - 增加了timestamp硬件
 - 利用timestamp硬件推动仿真执行
 - 可以利用系统的异步中断轮询功能
 - 可以多硬件交错异步执行
- 自带的测试属性：
 - 可以移植IO对应协议栈
 - 可以使用中断处理
 - 本地C执行速度是SV解析器执行速度的数倍
- 测试用例开发支持：
 - 提供协议栈/驱动框架的API
- 测试形态变化：
 - 社区软件皆可为测试激励
 - 接入Linux驱动框架
 - 接入RISC-V标准reference model

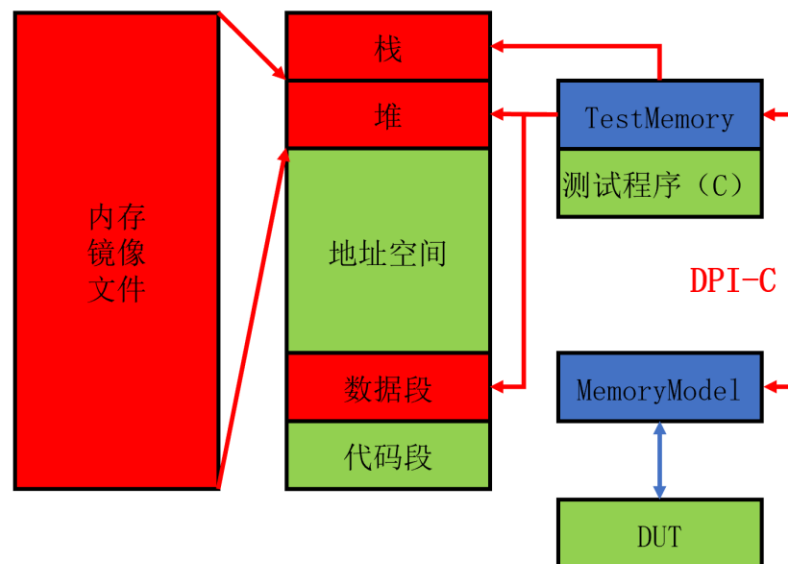
中断轮询在IP验证中使用

- CONFIG_SYS_IRQ
 - irq_register_vector: 注册中断处理例程（后台执行）
- CONFIG_SYS_NOIRQ
 - bh_register: 注册回调处理过程（前台执行）
 - irq_register_poller: 向轮询器申请轮询回调事件BH_POLL_IRQ，事件到达时执行中断处理例程
 - xxx_irq_init/xxx_poll_init
- IP验证中使用中断轮询
 - 利用sdfirm特有的中断轮询机制处理中断
 - 复用中断处理例程
 - 与异步状态机互动，实现完整的协议栈功能
 - 最终在IP验证环境中可使用带有中断处理的异步事件驱动



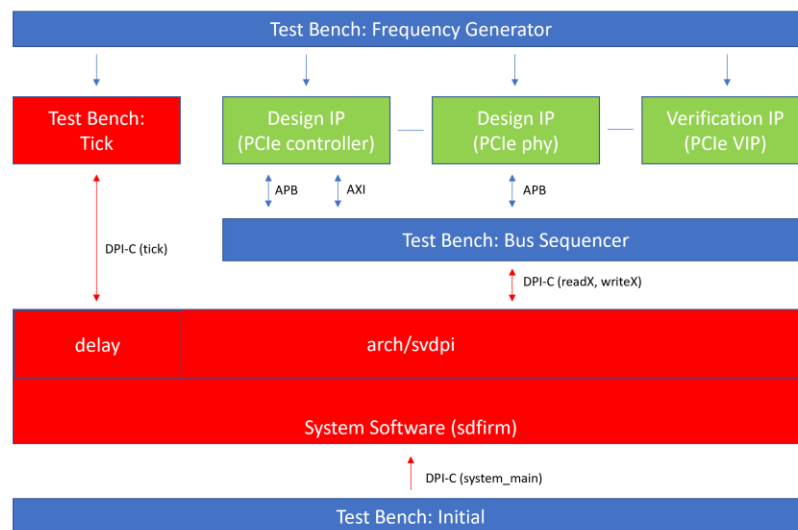
程序的指针处理

- IEEE1800规定C和SV之间的内存不共享
- 需求:
 - C直接修改的内存（数据和堆栈）需要被SV甚至RTL访问
 - `*(uint8_t *)ptr = a;`
 - `array[i] = b;`
 - 对C代码不作修改
- 方案:
 - 限制C内存map/unmap映射空间
 - C在映射空间上实现堆分配函数
 - 直接用程序内地址提供C侧内存访问的内存模型



仿真定时和仿真推进

- 驱动需要知道仿真执行的时间
 - SV提供返回仿真频率或某个总线频率基准的时间戳函数
- 进入C例程时，仿真停止
 - 长时间调用的C函数（wfi, __delay等）回调SV提供的仿真推进函数



驱动激励验证

- 使用sdfirm的SDHC驱动
 - CONFIG_MMCS=y
 - CONFIG_MMC_DEBUG=y
 - CONFIG_SD=y
- include/generic.h: 使用stub实现
- bh/irq: 使用sdfirm框架
 - CONFIG_SYS_NOIRQ
 - irq_register_poller
- 打印输入输出用仿真主机的C库函数实现
 - uart_hw_con_read: getch
 - uart_hw_con_poll: poll
 - uart_hw_con_write: putch
 - 不需要输入的话可以直接printf

```
cmd75 SELECT/DESELECT_CARD
sdhc: COMMAND: 000e
sdhc: ARGUMENT: 20000000
sdhc: RESPONSE:
00 00 06 20
event CMD_SUCCESS
state tran
op READ_BLOCKS
cmd17 READ_SINGLE_BLOCK
ARGUMENT(address): 0x00000200
sdhc: BLOCK_SIZE: 0200
sdhc: BLOCK_COUNT: 0001
sdhc: TRANSFER_MODE: 0012
sdhc: COMMAND: 000e
sdhc: ARGUMENT: 00000100
sdhc: RESPONSE:
00 00 09 20
```

```
sdhc: COMMAND: 000e
sdhc: ARGUMENT: 00000000
sdhc: RESPONSE:
00 00 09 20
event CMD_SUCCESS
cmd12 STOP_TRANSMISSION
sdhc: COMMAND: 000e
sdhc: ARGUMENT: 00000000
sdhc: RESPONSE:
00 00 09 20
event CMD_SUCCESS
Sfinish called from file "../vip/ahb_io.sv", line 86.
Sfinish at simulation time 271409ns
VCS Simulation Report
Time: 271409 ns
CPU Time: 2.140 seconds; Data structure size: 47.4Mb
Thu Apr 15 15:12:26 2021
```

sdfirm测试功能

sdfirm test capability

基准测试程序

- 支持的基准测试程序
 - dhrystone/linpack/coremark
- 支持SPIKE/Qemu
 - `make spike64_tb_defconfig`
 - `./scripts/run-spike.sh -p4 ./sdfirm`
- 执行方式: CPU Bench dIdT
 - `bench sync all dhrystone 1 0 1`
 - sync: 命令行完成后才可以执行下一条命令
 - async: 异步命令完成前可以执行其他测试命令
 - 可以指定CPU掩码, Cluster掩码, CPU Power Rail掩码
 - dIdT: 用于最高功耗的droop测试, 测试程序在所有CPU上同时开始执行来达到最高功耗, 在指定的测试周期period内不停重复执行, 在大于period的间隔期内CPU进入idle状态, repeats指idle唤醒后再次对齐时间点执行相同周期, 这样的测试轮次

```
LOWMAP: 0000000080000000 -> 0000000080000000: 00000000002d000 .text
LOWMAP: 000000008002d000 -> 000000008002d000: 000000000009000 .rodata
LOWMAP: 0000000080036000 -> 0000000080036000: 00000000000c000 .data
LOWMAP: 0000000080042000 -> 0000000080042000: 000000000004000 stack
LOWMAP: 000000008004a000 -> 000000008004a000: 000000007ffb6000 memory
SATP: 80000000008003a
SATP: 80000000008003e
smp: Allocating PERCPU area 000000008004a000(1).
smp: Bringing up CPU 0...
smp: Bringing up CPU 1...
SATP: 80000000008003a
SATP: 80000000008003e
smp: Bringing up CPU 2...
SATP: 80000000008003a
SATP: 80000000008003e
smp: Bringing up CPU 3...
SATP: 80000000008003a
SATP: 80000000008003e
Switch to the non-didt mode.
Begin time 134600, end time 190800
Number of runs: 5000
User time (us): 56200
VAX MIPS rating: 506
Begin time 134600, end time 190800
Number of runs: 5000
User time (us): 56200
VAX MIPS rating: 506
Begin time 134600, end time 190800
Number of runs: 5000
User time (us): 56200
VAX MIPS rating: 506
Begin time 134600, end time 190800
Number of runs: 5000
User time (us): 56200
VAX MIPS rating: 506
Command success 'bench - 0'
```

多核测试程序

- RISC-V多核一致性模型：
 - <https://github.com/litmus-tests/litmus-tests-riscv>
 - HEAD commit ID: 3782f858
- 支持sdfirm调度模型的litmus工具
 - <https://github.com/zetalog/herdtools7>
- 支持SPIKE/Qemu
 - 裸金属litmus测试, 适合前仿
 - 支持格式化打印输出, 支持MMU开启
 - `make spike64_litmus_defconfig`
 - `./scripts/run-spike.sh -p4 ./sdfirm`
- 批量执行脚本
 - `scripts/litmus.sh`
 - 批量执行所有测试, 自动执行单个测试, 批量生成所有测试, 批量执行测试子集, 测试的断点接续生成和执行
 - `./scripts/litmus.sh -e -p ./litmus-tests-riscv ISA03_2B_SIMPLE`

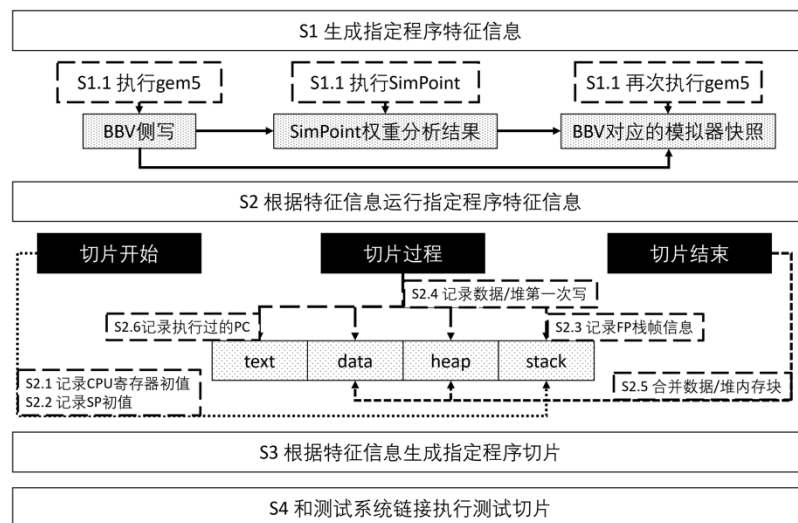
```
LOWMAP: 0000000080000000 -> 0000000080000000: 000000000014000 .text
LOWMAP: 0000000080014000 -> 0000000080014000: 000000000009000 .rodata
LOWMAP: 000000008001d000 -> 000000008001d000: 00000000000c000 .data
LOWMAP: 0000000080029000 -> 0000000080029000: 000000000004000 stack
LOWMAP: 0000000080031000 -> 0000000080031000: 000000007ffc000 memory
SATP: 800000000080021
SATP: 800000000080027
smp: Allocating PERCPU area 0000000080131000(1).
smp: Bringing up CPU 0...
smp: Bringing up CPU 1...
SATP: 800000000080021
SATP: 800000000080027
smp: Bringing up CPU 2...
SATP: 800000000080021
SATP: 800000000080027
smp: Bringing up CPU 3...
SATP: 800000000080021
SATP: 800000000080027
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Results for dummy.litmus %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RISCV dummy
"PodWW Rfe PodRR Fre"
{0:x5=1; 0:x6=x; 0:x7=y; 1:x6=y; 1:x8=x;}
P0      | P1      |
sw x5,0(x6) | lw x5,0(x6) ;
sw x5,0(x7) | lw x7,0(x8) ;

exists (1:x5=1 /\ 1:x7=0)
Generated assembler
Test dummy Allowed
Histogram (2 states)
4      :>1:x5=0; 1:x7=0;
4      :>1:x5=1; 1:x7=1;
No

Witnesses
Positive: 0, Negative: 8
Condition exists (1:x5=1 /\ 1:x7=0) is NOT validated
Hash=2939da84098a543efdbb91e30585ab71
Cycle=Rfe PodRR Fre PodWW
Relax dummy No
Safe=Rfe Fre PodWW PodRR
Generator=diy7 (version 7.51+4(dev))
Com=Rf Fr
Orig=PodWW Rfe PodRR Fre
Observation dummy Never 0 8
Time dummy 0.00
Test success.
```


切片测试程序

- 传统内存镜像执行不足：
 - 不能反映缓存的状态
 - 需要比较专业的转换工具
- 基于GEM5的解决方案：
 - CONFIG_GEM5
 - GEM5的google仓库中含有RISC-V反汇编支持
 - <https://github.com/zetalog/gem5/tree/gem5-next-all>
 - 仓库中含有ARM64反汇编支持
 - 有切片生成的整套实现
- Simpoint->可执行文件
 - 模拟器开始执行切片时跟踪
 - CPU寄存器
 - 栈：堆栈指针
 - 数据段/堆：
 - 模拟器执行切片过程中跟踪
 - 栈：帧
 - 数据段/堆：某地址第一次被读取的值
 - 代码段：跟踪跳转/调用的函数符号名称/函数符号首地址
 - 模拟器执行完切片：
 - 数据段/堆：合并临近地址内容形成离散内存列表
 - 生成程序：生成寄存器/内存加载和代码的汇编指令



嵌入式Linux测试系统

- OpenSBI的fw_payload模式合并S态到M态
 - CONFIG_SBI_PAYLOAD
 - CONFIG_SBI_PAYLOAD_PATH: Linux编译后的Image的路径
 - CONFIG_SBI_FDT_BUILTIN
- Linux的initramfs模式合并U态到S态
 - CONFIG_INITRAMFS_SOURCE
- 一键编译脚本:
 - scripts/linux/build_sdfirm_linux.sh
- 方便的测试功能:
 - BUILD_TINY: 尽可能裁剪linux/busybox
 - BUILD_LIB: 是否支持共享库
 - BUILD_SMP: 是否开启多核配置
 - TEST_EARLY: 早期可执行的测试, 例如benchmark、cpu2006、litmus
 - TEST_LATE: 晚期可执行的测试, 例如kvm、dmatest

```
SmartCore - RISC-V isa simulator (spike) Berkeley Bootloader
6.2.0-36-generic - 1.0.0.0

SDFIRM

Registering ecall legacy
Registering ecall base
Registering ecall time
Registering ecall ipi
Registering ecall rfence
Registering ecall hsm

OpenSBI v1.4 (Aug  7 2024 15:31:42)

OpenSBI

Platform Name           : RISC-V ISA simulator (spike)
Platform HART Features  : RV64ACDFHIMSU
Firmware Base           : 0x80000000
Firmware Size           : 168 KB
Runtime SBI Version     : 1.0
Platform Max HARTs      : 4
Firmware Max CPUs       : 4
Current Hart            : 0
Current CPU             : 0
Current Thread Pointer   : 0x0000000080026c00
Current Thread Stack     : 0x0000000080026000 - 0x0000000080027000
PMP0: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
PMP1: 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
[ 0.000000] Linux version 6.10.0-rc7-sdfirm+ (zhenglv@SW-Station) (
7 15:31:35 CST 2024
[ 0.000000] Machine model: ucbbar,spike-bare
[ 0.000000] SBI specification v1.0 detected
[ 0.000000] SBI implementation ID=0x1 Version=0x10004
[ 0.000000] SBI TIME extension detected
```

Litmus测试系统

- 需要的程序
 - sdfirm
 - linux
 - busybox
 - litmus_tests_riscv
 - 预先生成hw-tests-src重命名为litmus-4cores存放在litmus-tests-riscv下
 - 将run.c复制为run_all.c
- 测试编译脚本
 - spike64_bbl_defconfig
 - build_spike64_linux.sh
- 测试配置参数
 - BUILD_SMP=yes
 - LITMUS_CORES=4
 - LITMUS_UPDATE=yes
 - LITMUS_DUMP=no
 - LITMUS_ROOT=<path to litmus-tests-riscv>
 - TEST_EARLY=litmus
- 测试用例执行
 - ./sdfirm/scripts/run-spike.sh -p4
obj/sdfirm-riscv/sdfirm -u litmus.log

```
[ 0.465840] devtmpfs: mounted
[ 0.468300] Freeing unused kernel image (initmem) memory: 8920K
[ 0.468380] Run /sbin/init as init process
starting pid 45, tty '': '/etc/init.d/rcS'
mount: mounting none on /dev failed: Resource busy
===== Factory Address =====
192.168.10.1 255.255.255.0
=====
Running test 0 out of 60
Thu Jan 1 00:00:00 1970
% Results for tests/non-mixed-size/HAND/LB+amoadd-data-amoadd.rl+amoadd.aq-data-amoadd.litmus %
=====
RISCV LB+amoadd-data-amoadd.rl+amoadd-data-amoadd
{0:x3=x; 0:x6=y; 1:x3=y; 1:x6=x;}
P0
ori x2,x0,1      | P1
amoadd.w x1,x2,(x3) | amoadd.w aq x1,x2,(x3)
xor x4,x1,x1      | xor x4,x1,x1
ori x4,x4,1        | ori x4,x4,1
amoadd.w.rl x5,x4,(x6) | amoadd.w x5,x4,(x6)

exists (x=2 /\ y=2 /\ 0:x1=1 /\ 0:x5=0 /\ 1:x1=1 /\ 1:x5=0)
Generated assembler
#START _litmus_P0
ori t0,x0,1
amoadd.w a1,t0,(a5)
xor t6,a1,a1
ori t6,t6,1
amoadd.w.rl a0,t6,(a4)
#START _litmus_P1
ori t0,x0,1
amoadd.w.aq a1,t0,(a5)
xor t6,a1,a1
ori t6,t6,1
amoadd.w a0,t6,(a4)
Test LB+amoadd-data-amoadd.rl+amoadd.aq-data-amoadd Allowed
Histogram (2 states)
100000:>0:x1=1; 0:x5=1; 1:x1=0; 1:x5=0; x=2; y=2;
100000:>0:x1=0; 0:x5=0; 1:x1=1; 1:x5=1; x=2; y=2;
No
Witnesses
Positive: 0, Negative: 200000
Condition exists (x=2 /\ y=2 /\ 0:x1=1 /\ 0:x5=0 /\ 1:x1=1 /\ 1:x5=0) is NOT validated
Hash=02c052191eb1538ba909cf6fc7b1ead9
Observation LB+amoadd-data-amoadd.rl+amoadd.aq-data-amoadd Never 0 200000
Time LB+amoadd-data-amoadd.rl+amoadd.aq-data-amoadd 0.29
% Results for tests/non-mixed-size/HAND/LB+amoadd-data-amoadds.litmus %
=====
RISCV LB+amoadd-data-amoadds
```

SPEC2006测试系统

- 需要的程序
 - sdfirm
 - linux
 - busybox
 - cpu2006: SpecCPU 2006的CD解压目录
- 测试编译脚本
 - spike64_bbl_defconfig
 - build_spike64_linux.sh
- 测试配置参数
 - BUILD_TINY=yes
 - BUILD_LIB=yes
 - BUILD_SMP=yes
 - CPU2006_UPDATE=no
 - CPU2006_REPORT=no
 - CPU2006_REPORT=spec invoke
 - CPU2006_DATA=all
 - CPU2006_BENCHMARKS="401. bzip2"
 - TEST_EARLY=cpu2006

```
[ 0.042730] printk: bootconsole [ns16550a0] disabled
[ 1.600915] devtmpfs: mounted
[ 1.609250] Freeing unused kernel image (initmem) memory: 32936K
[ 1.621515] Run /sbin/init as init process
starting pid 39, tty '': '/etc/init.d/rcS'
mount: mounting none on /dev failed: Device or resource busy
===== Factory Address =====
192.168.10.1 255.255.255.0
=====
Running 401.bzip2...
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/chicken.jpg.err begin
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/chicken.jpg.err end
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/chicken.jpg.out begin
spec_init
Loading Input Data
Duplicating 652307 bytes
Duplicating 1304614 bytes
Duplicating 2609228 bytes
Duplicating 5218456 bytes
Duplicating 10436912 bytes
Duplicating 10583456 bytes
Input data 31457280 bytes in length
Compressing Input Data, level 5
Compressed data 31306985 bytes in length
Uncompressing Data
Uncompressed data 31457280 bytes in length
Uncompressed data compared correctly
Compressing Input Data, level 7
Compressed data 30427899 bytes in length
Uncompressing Data
Uncompressed data 31457280 bytes in length
Uncompressed data compared correctly
Compressing Input Data, level 9
Compressed data 25949409 bytes in length
Uncompressing Data
Uncompressed data 31457280 bytes in length
Uncompressed data compared correctly
Tested 30MB buffer: OK!
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/chicken.jpg.out end
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/input.combined.err begin
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/input.combined.err end
cpu2006: /opt/cpu2006/benchspec/CPU2006/401.bzip2/run/run_base_ref_riscv.0000/input.combined.out begin
spec_init
Loading Input Data
Duplicating 53779568 bytes
Duplicating 102156064 bytes
Input data 209715200 bytes in length
Compressing Input Data, level 5
Compressed data 44861975 bytes in length
Uncompressing Data
```

kvmtool 测试系统

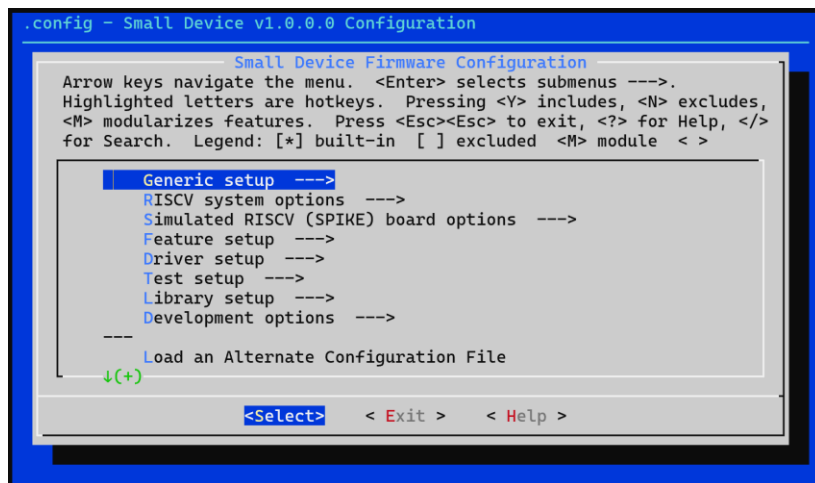
- 需要的程序
 - sdfirm
 - linux
 - busybox
 - kvmtool
 - dtc
- 测试编译脚本
 - spike64_bbl_defconfig
 - build_spike64_linux.sh
- 测试配置参数
 - BUILD_TINY=yes
 - BUILD_LIB=no
 - BUILD_SMP=yes
 - BUILD_NET=no
 - BUILD_STO=no
 - BUILD_KVM=yes
 - TEST_LATE=kvm

```
[ 0.114900] NET: Registered PF_PACKET protocol family
[ 0.130050] Legacy PMU implementation is available
[ 0.130090] clk: Disabling unused clocks
[ 0.575415] devtmpfs: mounted
[ 0.579215] Freeing unused kernel image (initmem) memory: 17264K
[ 0.579230] Run /sbin/init as init process
starting pid 27, tty '': '/etc/init.d/rcS'
mount: mounting none on /dev failed: Device or resource busy
===== Factory Address =====
192.168.10.1 255.255.255.0
=====
sdfirm: Configuring telnetd...[Success]
[ 0.623500] kvm [65]: hypervisor extension available
[ 0.623505] kvm [65]: using Sv57x4 G-stage page table format
[ 0.623515] kvm [65]: VMID 14 bits available
Info: # lkvm run -k ./Image -m 128 -c 1 --name guest-66
[ 0.000000] Linux version 6.10.0-rc7-sdfirm+ (zhenglv@SW-Station) (riscv64-u
7 15:30:26 CST 2024
[ 0.000000] Machine model: linux,dummy-virt
[ 0.000000] SBI specification v2.0 detected
[ 0.000000] SBI implementation ID=0x3 Version=0x60a00
[ 0.000000] SBI TIME extension detected
[ 0.000000] SBI IPI extension detected
[ 0.000000] SBI RFENCE extension detected
[ 0.000000] SBI SRST extension detected
[ 0.000000] SBI DBCN extension detected
[ 0.000000] earlycon: sbi0 at I/O port 0x0 (options '')
[ 0.000000] printk: legacy bootconsole [sbi0] enabled
[ 0.000000] Zone ranges:
[ 0.000000] DMA32 [mem 0x0000000080000000-0x0000000087fffffff]
[ 0.000000] Normal empty
[ 0.000000] Movable zone start for each node
[ 0.000000] Early memory node ranges
[ 0.000000] node 0: [mem 0x0000000080000000-0x0000000087fffffff]
[ 0.000000] Initmem setup node 0 [mem 0x0000000080000000-0x0000000087fffffff]
[ 0.000000] SBI HSM extension detected
[ 0.000000] Falling back to deprecated "riscv,isa"
[ 0.000000] riscv: base ISA extensions acdfim
[ 0.000000] riscv: ELF capabilities acdfim
[ 0.000000] percpu: Embedded 17 pages/cpu s38248 r0 d31384 u69632
```

测试软件编程

sdfirm test programming

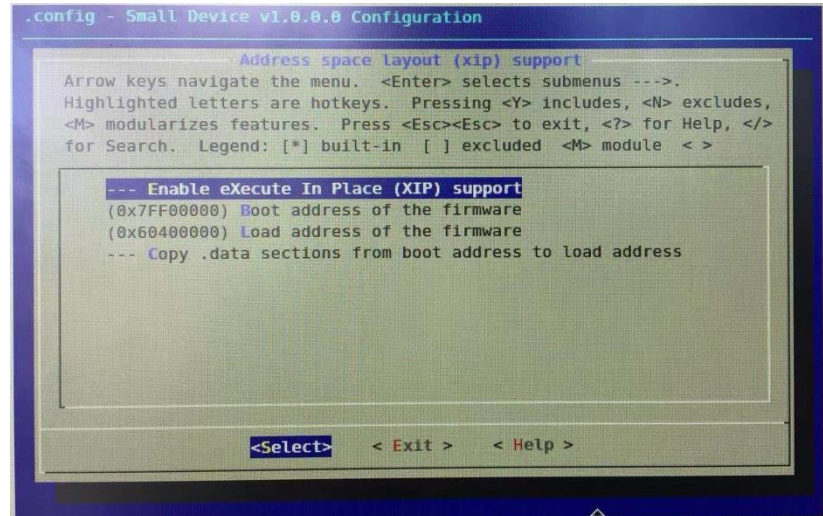
构建系统 (Kconfig/Kbuild)



- 下载和编译:
 - `git clone https://github.com/space-mit/sdfirm`
 - `export ARCH=riscv`
 - `export CROSS_COMPILE=riscv64-linux-`
 - `make menuconfig`
 - `make k1matrix_zsbl_defconfig`
 - `make k1matrix_bbl_defconfig`
 - `make clean`
 - `make`
- 配置菜单介绍
 - Generic setup
 - Program type
 - Console
 - GEM5
 - RISCv system options - arch
 - Xxx (spike) board options - mach
 - Feature setup - 子系统
 - Driver setup - 驱动
 - Library setup - libc, bitops, muldiv
 - Development options - 调试/仿真

二进制加载

- 内存排布
 - `arch/riscv/include/asm/mach/reg.h`:
 - `ROM_BASE/ROMEND`
 - `RAM_BASE/RAMEND`
 - `arch/riscv/common/sdfirm.ld.s`: 链接脚本
- Kconfig菜单:
 - Feature setup → Generic kernel features → Address space layout (xip) support
 - `CONFIG_XIP`: 引导加载时不复制 .text 段
 - `CONFIG_BFM`: 引导加载时不复制 .data 段
 - `CONFIG_BOOT_BASE`: 程序的入口地址, 例如 ROM 首地址
 - `CONFIG_LOAD_BASE`: .data 段的加载地址, 例如 RAM 首地址
- 仿真文件格式:
 - `CONFIG_VERILOG_DATA_WIDTH`: 控制 hex 文件每行位数



BSP基本功能

- **#include <target/clock.h>**
 - 时钟复位系统相关API：自动计算频率配置，自动解决时钟/复位依赖，自动解决仿真配置
 - include/asm/mach/clock.h：实现SoC手册定义的所有时钟/复位
 - include/dt-bindings/clock/sbi-clock-xxx.h：定义系统设计的频率
- 各种时钟驱动特点：
 - PLL：打开通常包括配置预设值频率并启用PLL，提供动态设置频率API
 - MUX：打开通常包括启用非osc_clk，并从osc_clk切到非osc_clk，一般提供源选择API
 - DIV：打开通常包括启用源时钟，解决依赖，提供动态设置频率API
 - CG：打开通常包括启用源时钟，解决依赖
 - RST：打开通常包括释放复位，解决依赖
- **#include <target/gpio.h>**
 - 管脚配置相关API：配置IO功能
 - include/asm/mach/gpio.h：实现SoC手册定义的所有管脚复用

API	说明
clk_enable	打开一个时钟，释放一个复位
clk_disable	关闭一个时钟
clk_set_frequency	动态设置某时钟频率
clk_get_frequency	获得某时钟当前设置的频率
clk_select_source	选择某时钟的输入源
gpio_config_pad	配置管脚上下拉，驱动力
gpio_config_mux	配置管脚复用

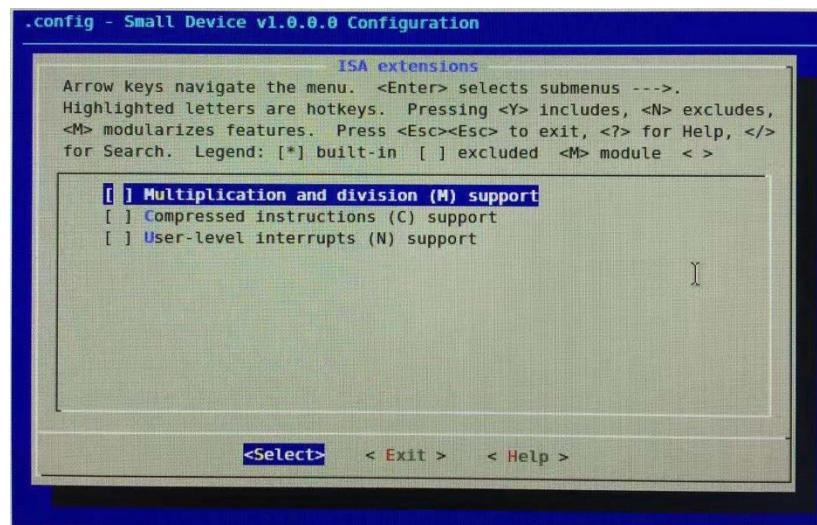
中断子系统

- 两种系统形态：
 - **CONFIG_SYS_IRQ**: 允许中断
 - **CONFIG_SYS_NOIRQ**: 只有中断轮询, 需要子系统的额外支持**BH_POLL_IRQ**
- 统一的API:
 - **#include <target/irq.h>**: ISR中断处理例程
 - **#include <target/bh.h>**: DSR任务队列例程
 - **include/asm/mach/irq.h**: 实现SoC手册定义的所有中断号

API	说明
irq_register_vector	注册ISR回调
irq_local_enable	开本地CPU全局中断
irq_local_disable	关本地CPU全局中断
irq_local_save	保存本地CPU全局中断
irq_local_restore	恢复本地CPU全局中断
irqc_enable_irq	启用外部中断
irqc_disable_irq	禁用外部中断
irqc_clear_irq	清除外部中断（不建议使用）
irqc_configure_irq	配置外部中断的触发、电平等信息
irqc_mask_irq	屏蔽外部中断（用于DSR处理开始）
irqc_unmask_irq	解屏蔽外部中断（用于DSR处理结束）
irqc_ack_irq	应答外部中断（建议使用）
bh_register_handler	注册DSR回调
bh_sync	类似于wait(), 转异步状态机为同步（需要console以外的保证所有DSR都没有事件）

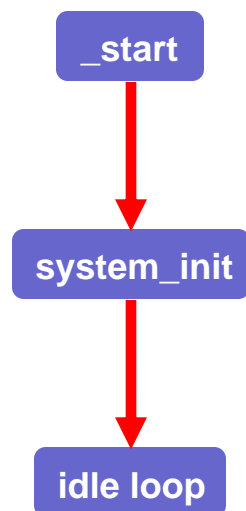
RISCV ISA选项

- arch/riscv/common/Kconfig: 定义CPU类型和ISA类型
 - CPU_E51/CPU_U54: sifive内核
 - CPU_R15CY/CPU_ORISCY: ETH zurich内核
 - CPU_C: 压缩指令支持
- menuconfig配置菜单中的程序形态
 - RISCV system options -> Base ISA
 - RISCV system options -> ISA extensions
 - RISCV system options -> Privileged software stack: 选择是AEE/SEE环境软件
 - Entry privilege level -> 程序启动时的权级
 - Exit privilege level -> 程序运行时的权级
 - Supervisor binary interface support -> Next privilege level: 下一跳程序的权级
 - 例如SBI软件: SEE、entry M、exit M、next S



启动过程

- arch/riscv/common/entry.S:
 - 初始化CPUs
 - 引导加载各个程序的段
 - 启用MMU - identity mapping
- init/main.c:
 - 准备早期的调试控制台（通过设备identity映射或者xo_clk驱动的串口，TSC）
 - 早期内存分配器bootmem allocator
 - 初始早期板级特定工作
 - 时钟/复位/管脚/中断初始化
 - 准备定时机制（tick/delay/timer）
 - 准备分页机制
 - 准备页分配器
 - 准备堆分配器
 - 注册命令行命令
 - 启用驱动（modules）
 - 进入等待中断的idle循环（bh_run_all）



编写测试代码

- 汇编测试用例写法:
 - `#include <target/compiler.h>`
 - `ENTRY()`: 定义汇编函数名
 - `ENDPROC()`: 结束一个汇编函数
 - `#ifdef __ASSEMBLY__`: 头文件中用于标识仅汇编器有效的代码
 - `#ifdef LINKER_SCRIPT`: 头文件中用于标识仅连接器有效的代码
- C测试用例的写法:
 - `#include <target/generic.h>`
 - `#include <target/xxx.h>`
 - `#include <stdio.h>/<string.h>`: 可以用一组小型的C库函数
- 头文件包含顺序
 - `#include <target/xxx.h>`
 - `-> #include <driver/xxx.h>`
 - `-> -> #ifdef CONFIG_ARCH_HAS_XXX`
 - `-> -> #include <asm/mach/xxx.h>`
 - `-> -> #define ARCH_HAVE_XXX 1`
 - `-> -> -> #include <asm/xxx.h>`
 - `-> -> -> #include <driver/xxx.h>`
- 使用编写的代码:
 - 有条件启用: 在Kconfig中准备选项`CONFIG_xxx`
 - 编译链接代码: 在Makefiles增加以下行:
`obj-$(CONFIG_xxx) += c_or_asm.rel`
 - 调用子系统代码: 在`system_init()`的恰当位置调用子系统初始化
- 定义测试命令行:
 - `DEFINE_COMMAND()`
 - `#include <target/cmdline.h>`

Thanks!

Thank you!