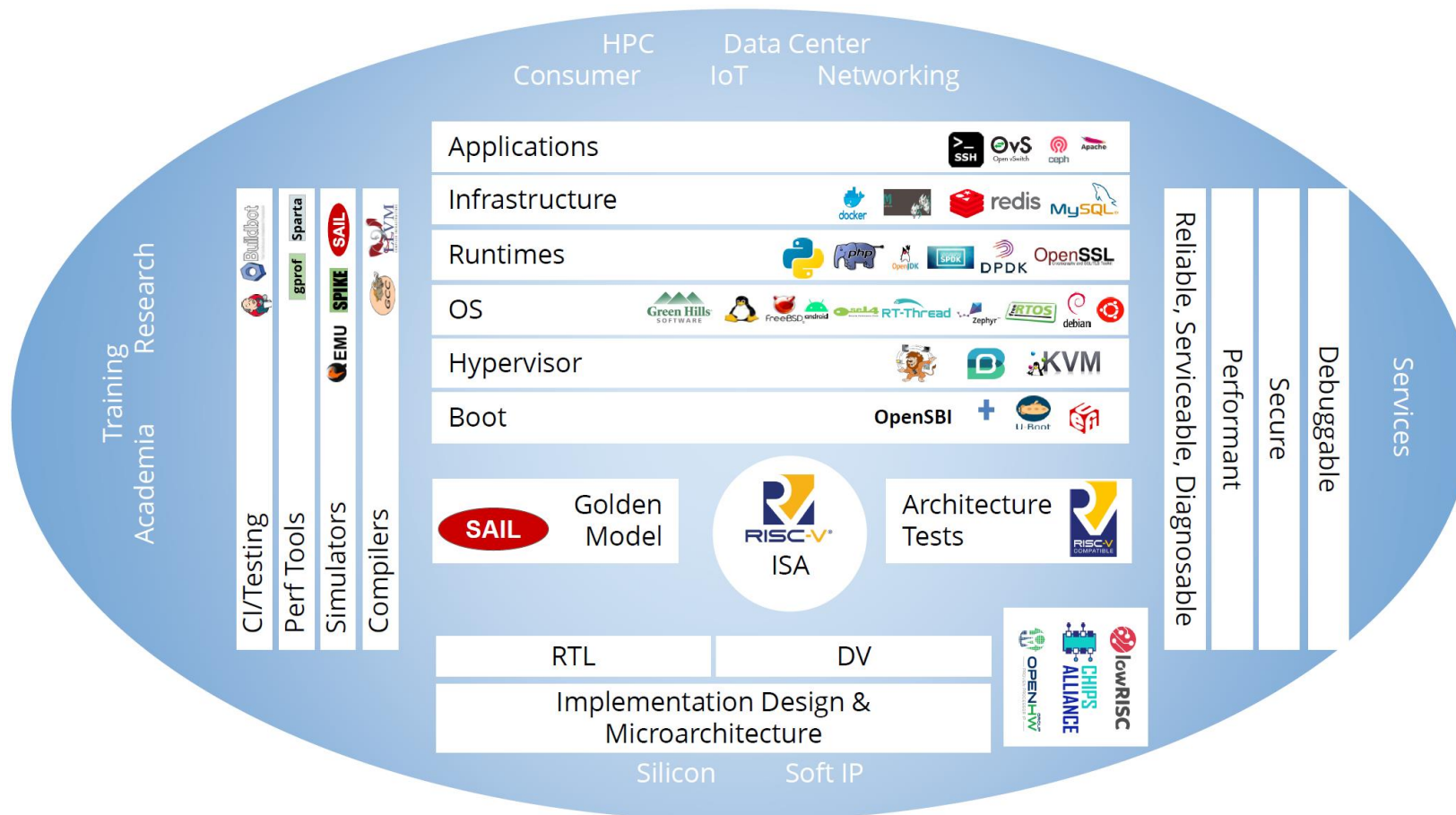




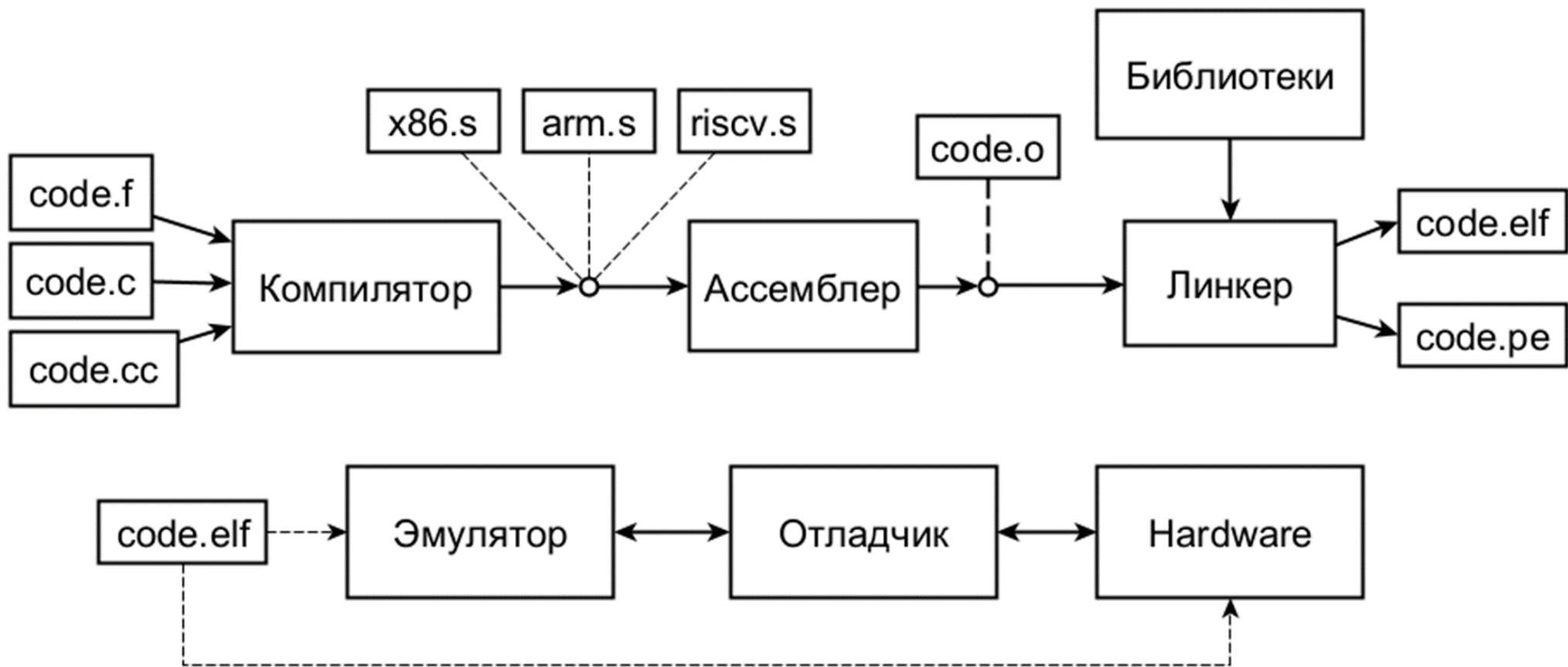
# Экосистема RISC-V

МФТИ  
Осень 2024

# Другая сторона айсберга



# Тулчейн – общий вид



# Кросс-компиляция

Основная терминология:

- **build** – машина, где происходит сборка компилятора.
- **host** – машина, где происходит сборка приложения.
- **target** – машина, где происходит запуск приложения.

Обычно для RISCV: build = host, host ≠ target

```
$ riscv64-linux-gnu-gcc-12 -march=rv64 file.c -O2 -S
```

Если host = target, то это нативная компиляция

Имя компилятора на хосте включает в себя target triple

```
<arch><sub>-<vendor>-<sys>-<env>
```

В рассматриваемом случае:


```
arch = riscv, sub = 64, vendor empty, sys = linux, env = gnu
```

# Hello, RISC-V!

```
$ cat hello.c
#include <stdio.h>
int main() { printf("Hello, world!\n"); }
```

```
$ uname -m
x86_64
```

```
$ riscv64-linux-gnu-gcc hello.c -static -o hello.x
```

```
$  hello.x
Hello, world!
```

# Hello, RISC-V!

```
$ cat hello.c
#include <stdio.h>
int main() { printf("Hello, world!\n"); }
```

```
$ uname -m
x86_64
```

```
$ riscv64-linux-gnu-gcc hello.c -static -o hello.x
```

```
$ qemu-riscv64 hello.x
Hello, world!
```

# Работа с платой

```
$ cat hello.c
#include <stdio.h>
int main() { printf("Hello, world!\n"); }
```

```
$ riscv64-linux-gnu-gcc hello.c -static -o hello.x
```

```
$ ssh user@board-ip mkdir ~/unique-name
```

```
$ scp hello.x user@board-ip:~/unique-name
```

```
$ ssh user@board-ip
```

```
$ uname -m
riscv
```

```
$ ~/hello.x
Hello, world!
```

# Концепция сжатых инструкций

- 4 байта на одну инструкцию это довольно много
- Большой размер одной инструкции
  - Создает излишнюю нагрузку на кэш инструкций
  - Увеличивает размер программ (критично для встраиваемых систем)
- Для решения этих проблем было создано C-расширение, которое добавляет «сжатые» (compressed) инструкции



# Концепция сжатых инструкций

- Из спецификации:
  - Typically, 50%-60% of the RISC-V instructions in a program can be replaced with RVC instructions, resulting in a 25%-30% code-size reduction.
- RVC uses a simple compression scheme that offers shorter 16-bit versions of common 32-bit RISC-V instructions when:
  - the immediate or address offset is small, or
  - one of the registers is the zero register (x0), the ABI link register (x1), or the ABI stack pointer (x2), or
  - the destination register and the first source register are identical, or
  - the registers used are the 8 most popular ones.

# Концепция сжатых инструкций

```
// -march=rv64i
000000000000000000 <elt>:
    0:    00259593 slli      a1, a1, 0x2
    4:    00a58533 add       a0, a1, a0
    8:    00852503 lw        a0, 8(a0)
```

```
// -march=rv64ic
000000000000000000 <elt>:
    0:    058a      c.slli    a1, a1, 0x2
    2:    952e      c.add     a0, a0, a1
    4:    4508      c.lw      a0, 8(a0)
```

# Собираем с расширениями

```
$ riscv64-linux-gnu-gcc file.c -march=rv64i  
-mabi=lp64 -static -O2 -o app
```

```
$ riscv64-linux-gnu-gcc file.c -march=rv64im  
-mabi=lp64 -static -O2 -o app
```

```
$ riscv64-linux-gnu-gcc file.c -march=rv64imf  
-mabi=lp64f -static -O2 -o app
```

```
$ riscv64-linux-gnu-gcc file.c -march=rv64imfd  
-mabi=lp64d -static -O2 -o app
```

```
$ riscv64-linux-gnu-gcc file.c -march=rv64gc  
-mabi=lp64d -static -O2 -o app
```

# Пришло время отладки

```
$ riscv64-linux-gnu-gcc buggy-sort.c -static -o  
buggy-sort
```

```
$ ./buggy-sort
```

[buggy-sort.c sources](#)



# Кросс-отладка

По аналогии с кросс-компиляцией:

- **build** – машина, где происходит сборка отладчика
- **host** – машина, где происходит запуск фронтенда отладчика
- **target** – машина, где происходит запуск отлаживаемого приложения

Host:

```
$ riscv64-linux-gnu-gdb buggy-sort  
> target extended-remote <board-ip>:<port>
```

Target:

```
$ gdbserver :<port> <program> [ args ... ]
```

Так как код выполняется на **target** машине, то и *весь вывод приложения* показывается в консоли **target**'а

# Всегда ли нужна плата?

- `qemu-riscv32` и `qemu-riscv64`, из `sc-dt` позволяют запускать RISC-V приложения
- В Syntacore Development Toolkit так же есть `qemu-system-32` и `qemu-system-64`, которые позволяют симулировать систему целиком и фактически являются виртуальной машиной
- Например, в `system qemu` можно загрузить линукс и протестировать кросс отладку, пока плата еще не готова

# To be continued ...

На следующем занятии узнаем как сделать свой RISC-V процессор и

- Как ведется разработка программного обеспечения для процессора, которого нет
- Когда решают, что пора оформлять заказ на производство
- Почему первый блин всегда комом
- Как верифицируют процессоры

# Список литературы

- The RISC-V Instruction Set Manual Volume I Unprivileged Architecture Version 20240411 // Chapter 26. "C" Extension for Compressed Instructions, Version 2.0
- Hennessy J. L., Patterson D. A. Computer architecture: a quantitative approach. – Morgan kaufmann, 2017.
- Toolchain & pony. Константин Владимиров:  
<https://youtube.com/playlist?list=PL3BR09unfgchnnggx7IJuSU57mxjMhrSaq>
- GDB Cheat Sheet:  
<https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf>