Chapter-2

(How do we get a RISC-V instruction set from a high-level code?)

RISC-V → instruction set architecture.

Reduced Instruction Set Computer

an abstract interface between the hardware and the lowest level software.

RISC-V ISA has a modular design.

Base + Optional Extension

ISA

Unit	
Double Word	= 64 bits
Word	= 32 bits
Half Word	= 16 bits
Byte	= 8 bits

Register:

⇒ Used for frequently accessed data. Registers are faster than regular memory.

⇒ RISC-V has 32, 64-bit registers.] 64 bit architecture ⇒ handling 64 bit data at a time.

↓

Follows Design Principle 2 ⇒ Smaller is faster

↳ The clock period of the system could be limited by many different things, Register File is also one of those factors.

Hence, computer architects pick the size of the register file very carefully. 32 × 64 bit is a small enough register file that is not a limiting factor for our systems.

RISC-V Register Details:

Theoretically you could store any information in any register but architects/programmers have decided to follow a convention, in which they use certain registers for certain purposes.

↓

increases readability

Register Num.	Functionality
x0	Constant value 0 ⇒ holds constant value 0.
x1	Return address ⇒ to store the return address that you should return to after a function call.
x2	Stack pointer ⇒ stores an address in memory with the top of the stack.
x3	Global pointer
X4	Thread pointer
x5-7 ✓✓	Temporaries
x8	~~Saved register~~ / Frame pointer
x9 ✓	Saved register ⇒ to store variables.
x10-11	Function arguments / return values] for values passed to a function and values returned from a function.
x12-17	Function arguments
x18-27 ✓	Saved registers
x28-31 ✓✓	Temporaries ⇒ For holding temp. values

|| ⇒ Saved reg.

7 ⇒ Temporary reg.

Operands: part of instruction that contains the data to acted on

Based on the physical location

- Register operand (small capacity + faster access time)
- Memory operand (large capacity + longer access time)
- Constant / immediates (physically in the instruction itself)

or, the memory location of the data in a register.

Register Operand: Arithmetic instructions use register operands.

$f = (g+h) - (i+j);$

↳ Only one operation is performed per RISC-V instruction.

Hence, compiler must break this statement down to several parts.

RISC-V code:

g+h is a step towards our actual answer. Hence, stored in the temp regs. one of

```
add  x5, x20, x21
add  x6, x22, x23
sub  x19, x5, x6
```

same explanation as the prev. one

1 codeline divided into multi-ple instruction lines

$f \Rightarrow x_{19}$ ✓
$g \Rightarrow x_{20}$
$h \Rightarrow x_{21}$
$i \Rightarrow x_{22}$
$j \Rightarrow x_{23}$

Add / Sub instruction syntax:

Instruction name

```
add ____ , ____ , ____
    dest.  Source1  Source2
```

⇒ dest. = source1 + source2

```
sub ____ , ____ , ____
    dest.  Source1  Source2
```

⇒ dest. = source1 − source2

always 3 register operands

unique memory