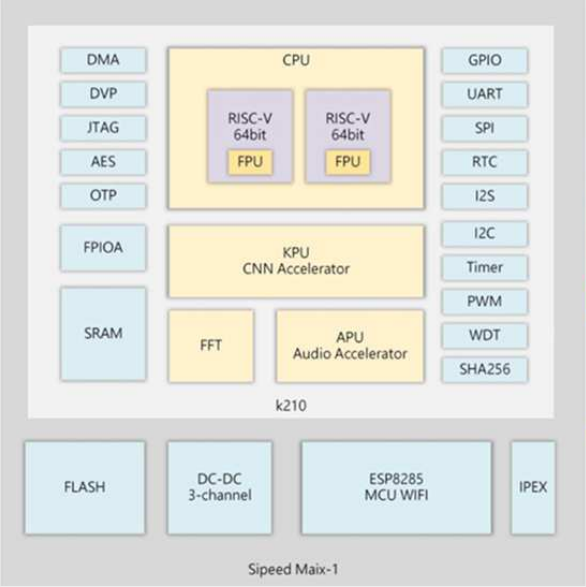
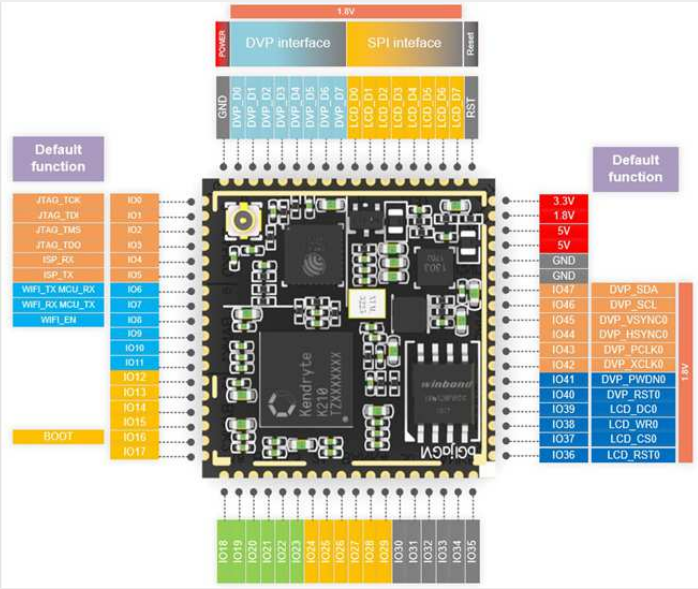
Risc-V Overzicht:

- 32-bit, 64-bit of 128-bit woordbreedte
- Variabele instructielengte (per 16-bits)
- 16 of 32 Registers (een nul register)
- Load-and-Store
- Compare-and-branch
- Little-endian

Risc-V structuur:





Risc-V instructie formaat:

32-bit RISC-V Instruction Formats

Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register/register	funct7							rs2					rs1					funct3			rd					opcode						
Immediate	imm[11:0]												rs1					funct3			rd					opcode						
Upper Immediate	imm[31:12]																				rd					opcode						
Store	imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						
Branch	[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				[11]	opcode						
Jump	[20]	imm[10:1]										[11]	imm[19:12]								rd					opcode						

- **opcode (7 bit):** partially specifies which of the 6 types of *instruction formats*
- **funct7 + funct3 (10 bit):** combined with **opcode**, these two fields describe what operation to perform
- **rs1 (5 bit):** specifies register containing first operand
- **rs2 (5 bit):** specifies second register operand
- **rd (5 bit):**: Destination register specifies register which will receive result of computation

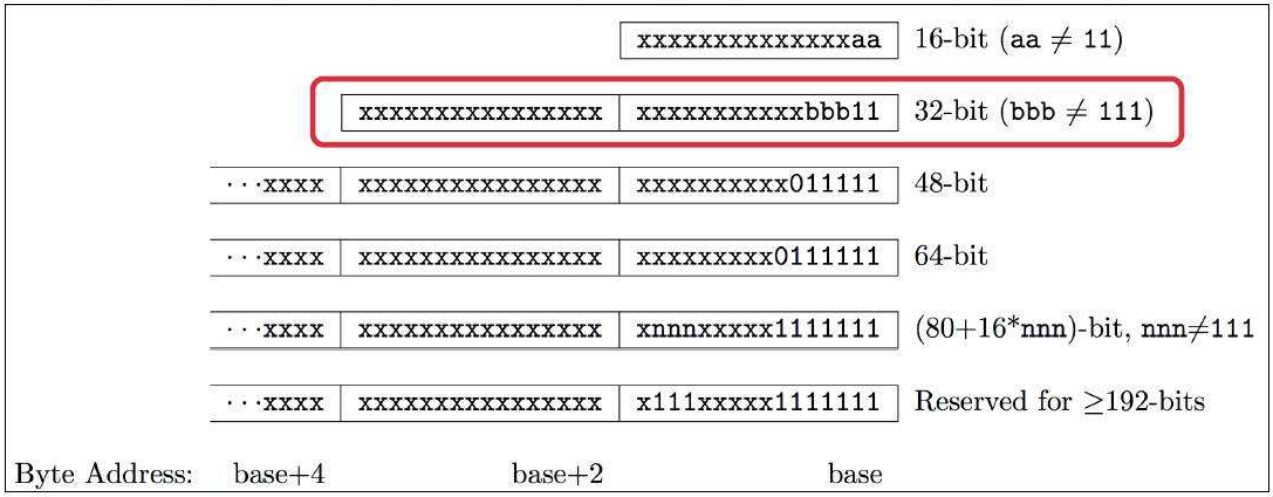
Register set:

Register name	Symbolic name	Description	Owner
32 integer registers			
x0	Zero	Always zero	-
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	-
x4	tp	Thread pointer	-
x5	t0	Temporary / alternate return address	Caller
x6–7	t1–2	Temporary	Caller
x8	s0/fp	Saved register / frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function argument / return value	Caller
x12–17	a2–7	Function argument	Caller
x18–27	s2–11	Saved register	Callee
x28–31	t3–6	Temporary	Caller
32 floating-point extension registers			
f0–7	ft0–7	Floating-point temporaries	Caller
f8–9	fs0–1	Floating-point saved registers	Callee
f10–11	fa0–1	Floating-point arguments/return values	Caller
f12–17	fa2–7	Floating-point arguments	Caller
f18–27	fs2–11	Floating-point saved registers	Callee
f28–31	ft8–11	Floating-point temporaries	Caller

Risc-V instructie codering:

RISC-V Hybrid Instruction Encoding

- 16, 32, 48, 64 … bits length encoding
- Base instruction set (RV32) always has fixed 32-bit instructions lowest two bits = 11_2
- All branches and jumps have targets at 16-bit granularity (even in base ISA where all instructions are fixed 32 bits



De Risc-V basis is een 32-bits instructie formaat!

- Bij 16-bit instructies zijn de aa-bits niet 11
- Bij 32-bit instructies de laagste twee bits 11 en de bbb-bits ongelijk aan 111
- Bij 48-bits instructies zijn de laagste 6-bits 011111
- Bij 64-bits instructies zijn de laagste 7-bits 0111111
- Etc.

RV32IMAC instructieset:

Ook wel Bumblebee genoemd, de basis is de 32-bits instructieset RV32I.

`I` – 47 stuks 32-bit integer core instructies
`M` – 8 stuks vermenigvuldig en deel instructies
`A` – 11 stuks multi-core communicatie instructies
`C` – 49 stuks 16-bit instructieset

De vermenigvuldiging gaat in een kloktik, de deling is afhankelijk van de data.

De 16-bits instructies hebben dezelfde functionaliteit als de 32-bits. De beperkingen zijn, het adres bereik va een sprong of de getal constante. De bruikbare registers en of het samenvoegen van het eerste source register met de destination!

Integer Register-Immediate Instructions

31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM

32-bit ANDI

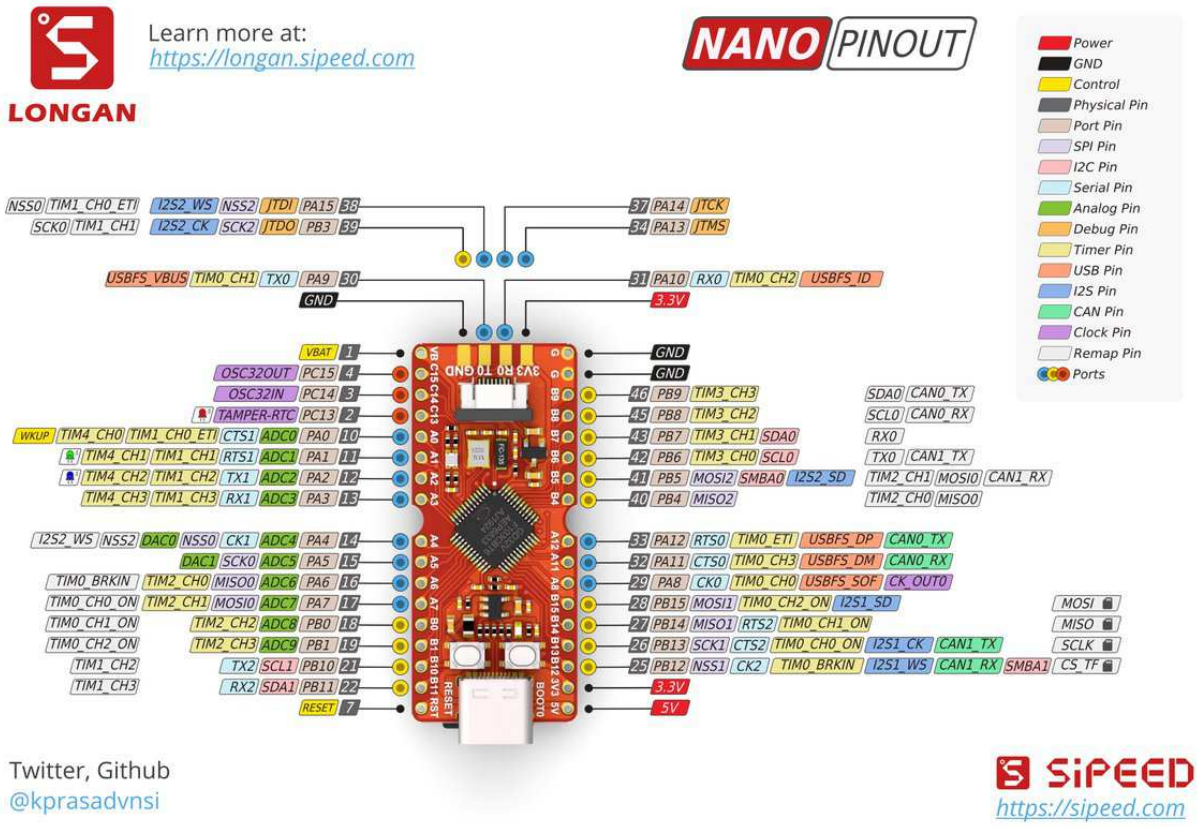
15 13	12	11 10	9 7	6 2	1 0
funct3	imm[5]	funct2	rd′/rs1′	imm[4:0]	op
3	1	2	3	5	2
C.ANDI	imm[5]	C.ANDI	dest	imm[4:0]	C1

16-bit ANDI

Risc-V processors:

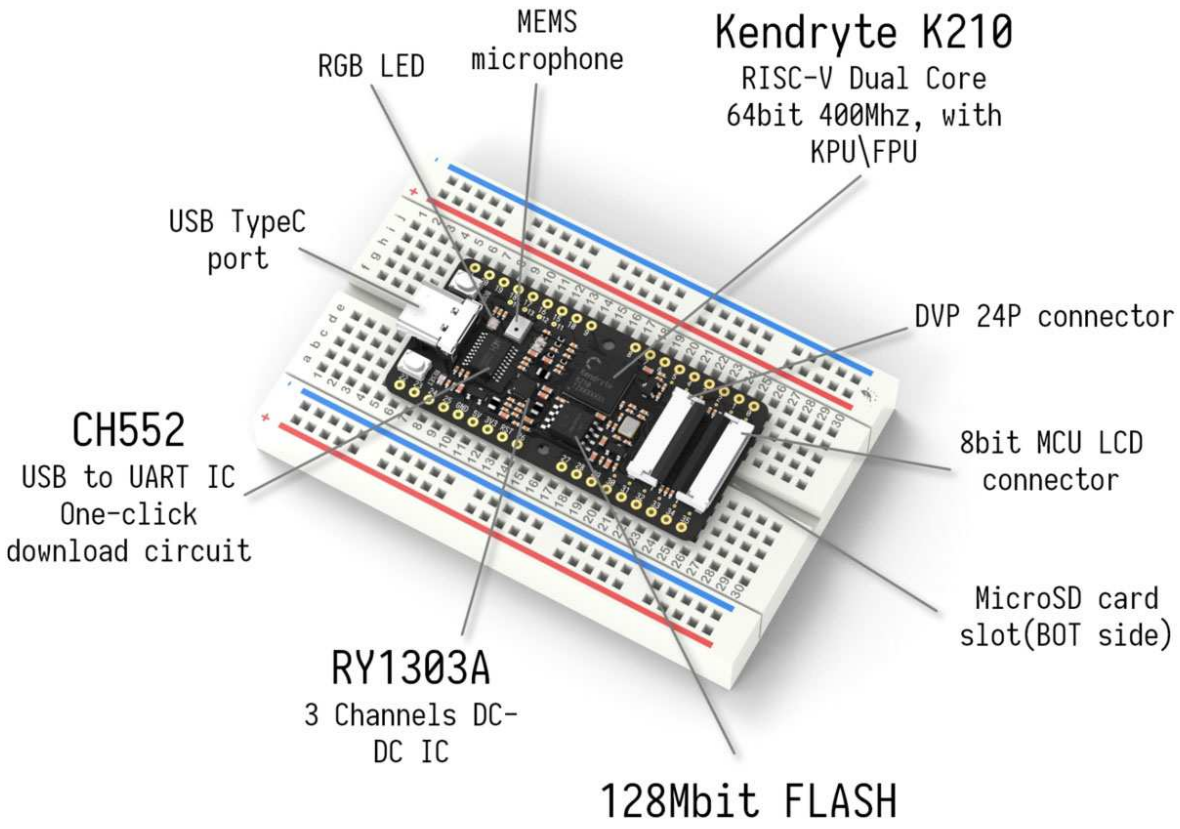
- **SiFive 64-bit U74**, op 2,5 Ghz
- **Xuantie 910**, met 16 cores op 2,5 Ghz 12nm struct.
- **Kendryte K210**, 64-bit 400MHz, 28nm struct, dualcore, 16 Mbyte ROM, 8 MByte SRAM, heel veel hardware..
- **GD32VF103**, 128 kByte Flash, 32 kByte SRAM, 10-timers, RTC, 5 UART's, 2xI2C, 3xSPI, 16XADC (12-bit) 2xDAC, CAN, 51xI/O, USB, etc.

Sipeed Nano:



Met **GD32VF103**

Sipeed MAix Bit:



Met **Kendryte K210**

About the performance:

- The K210’s KPU has 0.8 TFLOPS.
- The NVIDIA Jetson Nano with 128 CUDA unit GPU's has a power of 0.47 TFLOPS.
- Raspberry Pi 4 has less than 0.1 TFLOPS.

Forth code voorbeelden ITC

```
DOCOLON:
        push x16        \ IP
        mv x16,x17      \ W->IP
DO_NEXT:
        lw x17, 0(x16)  \ @IP -> W
        addi x16,x16,4  \ INC IP
DO_EXECUTE:
        lw x10, 0(x17)  \ @W, addr exec code
        addi x17,x17,4  \ INC W, points to PFA
        jalr zero,x10,0 \ jump to code


# ------------------------------------------
  CODEWORD "1-", 1MINUS # ( u -- u-1 )
# ------------------------------------------
  addi x3, x3, -1
  j DO_NEXT

# ------------------------------------------
  CODEWORD "+", PLUS # ( x1 x2 -- x1+x2 )
# ------------------------------------------
  lw x5, 0(x4)
  addi x4, x4, 4
  add x3, x5, x3
  j DO_NEXT
```

Forth code voorbeelden STC

```
# -----------------------------------------
# "1-" # ( u -- u-1 )
# Subtracts one from the top of the stack.
# -----------------------------------------
  addi x8, x8, -1
  ret

# -----------------------------------------
# "+" # ( x1 x2 -- x1+x2 )
# Adds x1 and x2.
# -----------------------------------------
  lw x15, 0(x9)
  addi x9, x9, 4
  add x8, x15, x8
  ret
```