

PULP RISC-V ARA

Dr. Girish H

Professor

Department of ECE

Cambridge Institute of Technology

&

Kavinesh

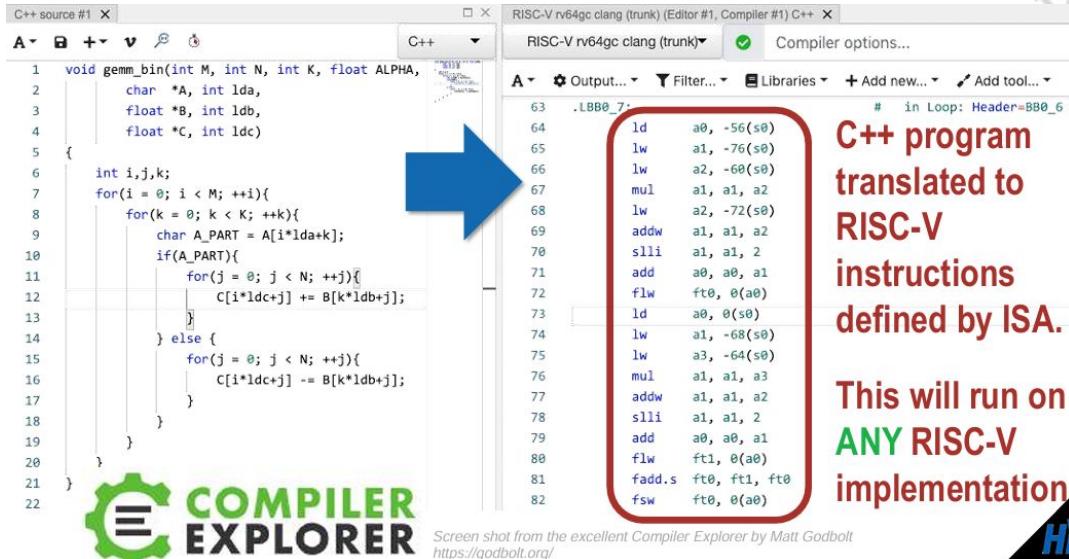
Research Staff

CCCIR

Cambridge Institute of Technology

RISC-V ISA is divided into extensions

ISA defines the instructions that processor uses



A blue arrow points from the C++ source code on the left to the RISC-V assembly output on the right.

C++ source #1 x

```
1 void gemm_bin(int M, int N, int K, float ALPHA,
2     char *A, int lda,
3     float *B, int ldb,
4     float *C, int ldc)
5 {
6     int i,j,k;
7     for(i = 0; i < M; ++i){
8         for(k = 0; k < K; ++k){
9             char A_PART = A[i*lda+k];
10            if(A_PART){
11                for(j = 0; j < N; ++j){
12                    C[i*ldc+j] += B[k*ldb+j];
13                }
14            } else {
15                for(j = 0; j < N; ++j){
16                    C[i*ldc+j] -= B[k*ldb+j];
17                }
18            }
19        }
20    }
21 }
```

COMPILER EXPLORER

Screen shot from the excellent Compiler Explorer by Matt Godbolt
<https://godbolt.org/>

RISC-V rv64gc clang (trunk) (Editor #1, Compiler #1) C++ x

RISC-V rv64gc clang (trunk) Compiler options...

```
63 .LBB0_7:
64    ld    a0, -56($0)
65    lw    a1, -76($0)
66    lw    a2, -60($0)
67    mul   a1, a1, a2
68    lw    a2, -72($0)
69    addw  a1, a1, a2
70    slli  a1, a1, 2
71    add   a0, a0, a1
72    flw   ft0, 0(a0)
73    ld    a0, 0($0)
74    lw    a1, -68($0)
75    lw    a3, -64($0)
76    mul   a1, a1, a3
77    addw  a1, a1, a2
78    slli  a1, a1, 2
79    add   a0, a0, a1
80    flw   ft1, 0(a0)
81    fadd.s ft0, ft1, ft0
82    fsw   ft0, 0(a0)
```

C++ program translated to RISC-V instructions defined by ISA.

This will run on ANY RISC-V implementation

- I** Integer instructions (frozen)
- E** Reduced number of registers
- M** Multiplication and Division (frozen)
- A** Atomic instructions (frozen)
- F** Single-Precision Floating-Point (frozen)
- D** Double-Precision Floating-Point (frozen)
- C** Compressed Instructions (frozen)

Reduced Instruction Set: all in one page

Optional Compressed (26-bit) Instruction Extension: RVC											
Category		Name	Type	RVC Encoding							
Load	Load Word	LD	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Load	Load Halfword	LHD	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Load	Load Word	LDW	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Load	Load Halfword	LWD	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Stores	Store Byte	SB	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Stores	Store Halfword	SH	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Stores	Store Word	SW	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Left	SL	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Left Immediate	SLI	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Right	SR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Right Immediate	SRI	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Right Arithmetic	SRA	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Arithmetics	Add	ADD	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Arithmetics	Add Immediate	ADDI	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Arithmetics	Subtract	SUB	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Load Upper Immediate	LDU	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm	
Add Upper Immediate	ADDU	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm	
Logical	XOR	XOR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
XOR Immediate	XORI	XOR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
GR	GR	GR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
OR Immediate	ORI	GR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
AND Immediate	ANDI	GR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Compare	Set =	SET	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Compare	Set < Immediate	SETL	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Compare	Set < Imm Unsigned	SETLU	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch	BR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch a	BRa	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch b	BRb	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch c < Unsigned	BRcu	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch d < Unsigned	BRdu	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Jump & Link	JBL	JBL	RD	rd,r1,imm	rd	r1	imm		rd	imm	imm
Jump & Link Register	JBLR	JBLR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
System	System CALL	SCALL	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
System	System BREAK	SBREAK	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
System	Sync Instr & Break	SINSTR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Counters	Read CYCLIC COUNT	RCCYCLC	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Counters	Read CYCLIC TIME	RCCYCLT	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Counters	Read TIME	RTIME	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Counters	Read TIME upper half	RTIMEH	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Counters	Read DISTRA	RDISTRA	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Counters	Read DISTRA upper half	RDISTRAH	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Left	SL	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Left Register	SLR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Right	SR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Shifts	Shift Right Register	SSR	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch a	BRa	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Branches	Branch b	BRb	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Jump	Jump	JMP	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Jump & Link Register	JMPRL	JMPRL	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
System	System TRAP	STRAP	RD	rd,r1,r2,imm	rd	r1	r2	imm	rd	imm	imm
Compressed Instructions (C)											
Privilege Mode											
Multiply/Divide (M)											
Atomic Extensions (A)											
PIC32 - C Calling Convention											
Register	ABI Name	Save	Description								
A0	SP	zero	Hard-wired zero								
A1	BP	bp	Base pointer								
A2	TP	tp	Stack pointer								
A3	DP	dp	Global pointer								
A4	IP	ip	Thread pointer								
A5	TPR	tp	Temporary								
A6	SPR	sp	Register-frame pointer								
A7	SR	sr	Saved register								
A8	AR	ar	Function arguments/return values								
A9	VR	vr	Function arguments								
A10	VRP	vpr	Temporary								
A11	SPR	sp	FP temporaries								
A12	SPR	sp	FP saved registers								
A13	SPR	sp	FP arguments/return values								
A14	SPR	sp	FP arguments								
A15	SPR	sp	FP temporary								
A16	SPR	sp	FP arguments/return values								
A17	SPR	sp	FP arguments								
A18	SPR	sp	FP temporary								
A19	SPR	sp	FP arguments/return values								
A20	SPR	sp	FP arguments								
A21	SPR	sp	FP temporary								
A22	SPR	sp	FP arguments/return values								
A23	SPR	sp	FP arguments								
A24	SPR	sp	FP temporary								
A25	SPR	sp	FP arguments/return values								
A26	SPR	sp	FP arguments								
A27	SPR	sp	FP temporary								
A28	SPR	sp	FP arguments/return values								
A29	SPR	sp	FP arguments								
A30	SPR	sp	FP temporary								
A31	SPR	sp	FP arguments/return values								
Floating Point Extensions											
Register	Read Status	RS	rd								
Register	Swap Status Reg	RSR	rd								
Register	Swap Rounding Mode	SR	rd								
Register	Swap Flags	SF	rd								
Register	Swap Rounding Mode Swap	SRSWAP	rd								
Register	Swap Status Swap	SSWAP	rd								
Register	Swap Flags Swap	FSWAP	rd								
Register	Swap Rounding Mode Swap Status	SRSSWAP	rd								
Register	Swap Status Swap Flags	SSFSWAP	rd								
Register	Swap Rounding Mode Swap Status Swap	SRSSFSWAP	rd								
Register	Swap Status Swap Flags Swap	SSFSWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags	SRSSFSWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status Swap	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status Swap	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status Swap	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status Swap	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status Swap Rounding Mode	SSFSRWAP	rd								
Register	Swap Rounding Mode Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status Swap	SRSSFSRWAP	rd								
Register	Swap Status Swap Flags Swap Rounding Mode Swap Status Swap Rounding Mode Swap Status Swap Rounding Mode Swap	SSFSRWAP	rd								

PULP Parallel Ultra Low Power

- It is an open-source platform for designing energy-efficient, parallel computing systems.
- It is based on the RISC-V instruction set architecture (ISA)
- The PULP platform is designed to support efficient parallel processing.



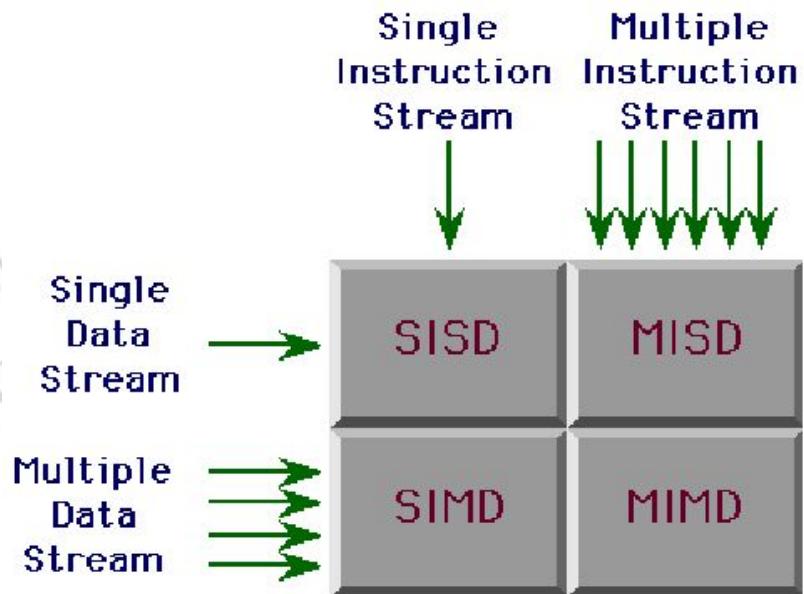
Energy
Efficient



Classification to achieve parallel computing

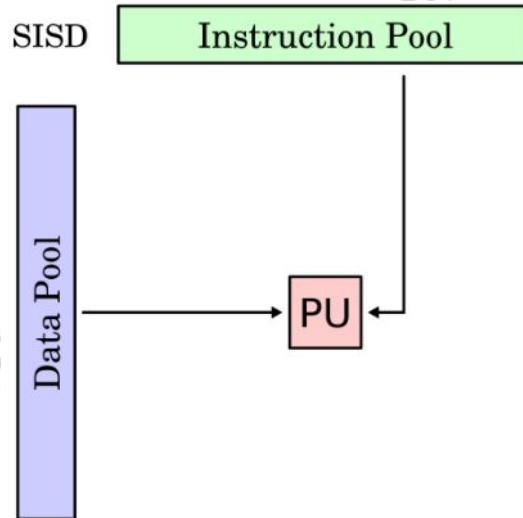
Flynn's Taxonomy

- Proposed by Michael J. Flynn in 1966
- Overview of the design space of parallel computers
- Characterization according to the data flow and the control flow



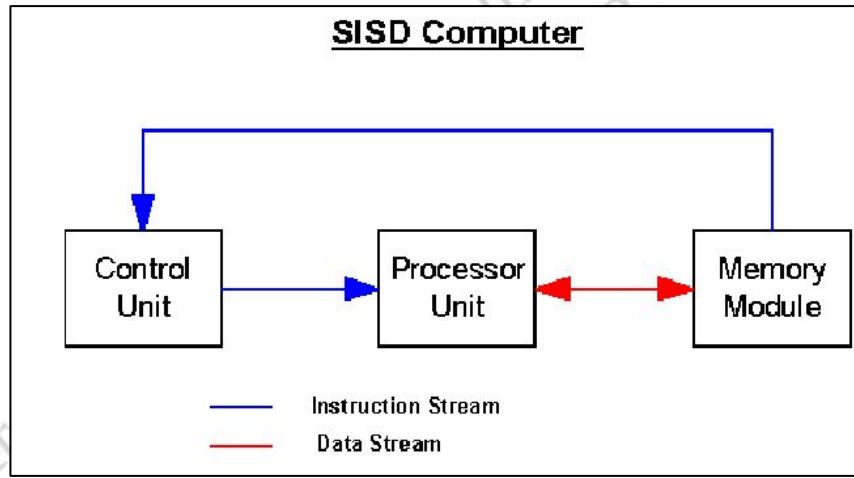
Single-Instruction, Single-Data

- One processing element
- It has access to a single program and a single data source

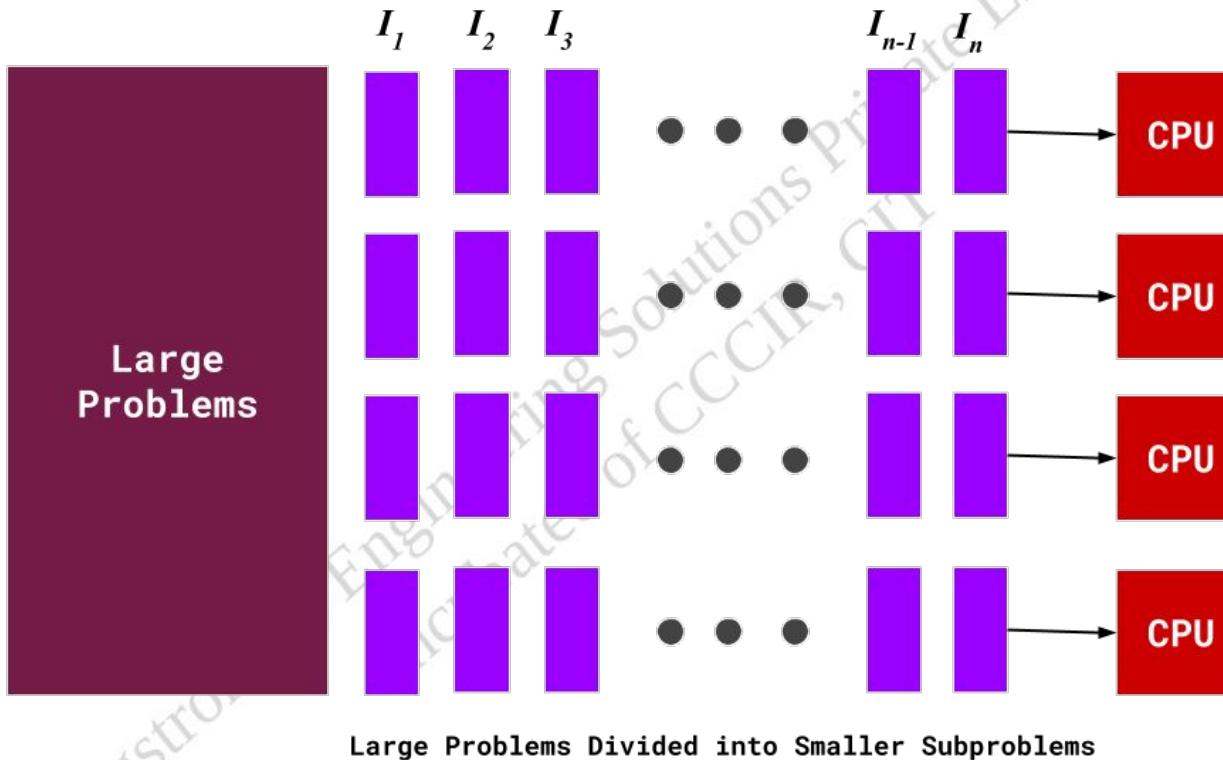


Single Instruction, Single Data Stream - SISD

- The PE fetches instructions and data from the main memory processes the data as per the instructions and sends the results to the main memory after processing has been completed

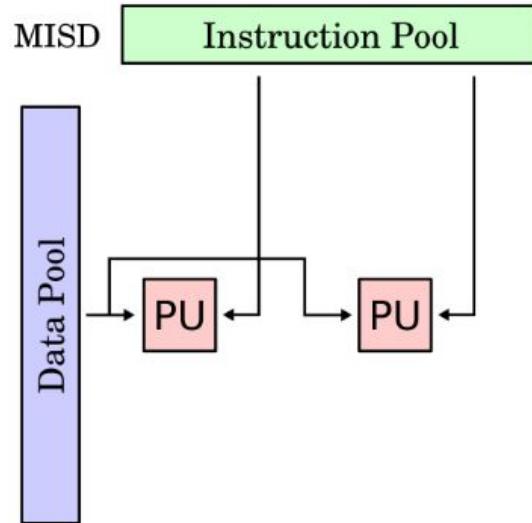


SISD is inefficient to solve for large data



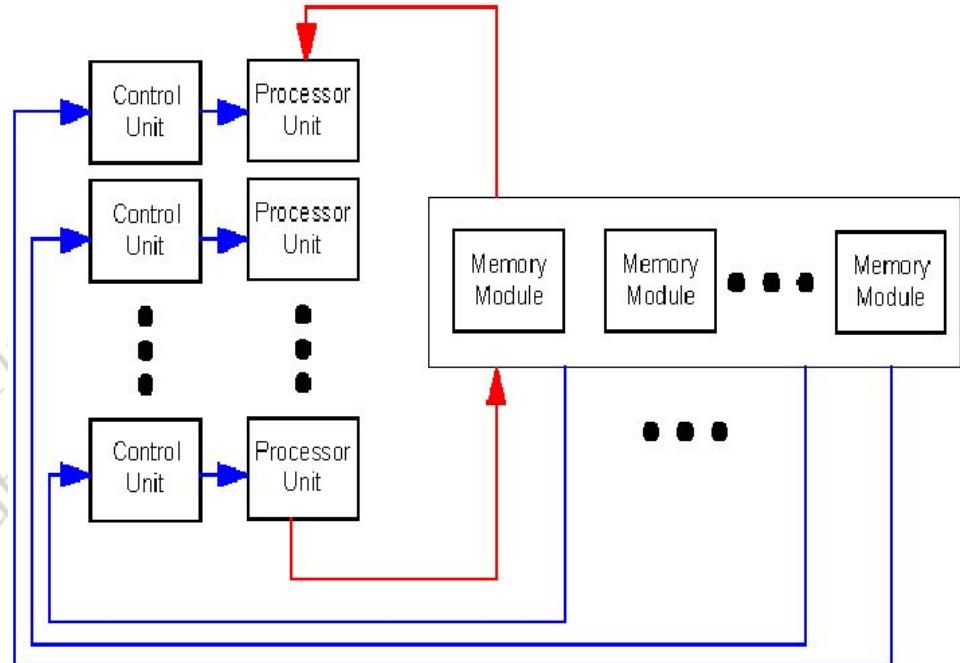
Multiple-Instructions, Single-Data

- Multiple processing elements
- Each one executes a different program on the same data



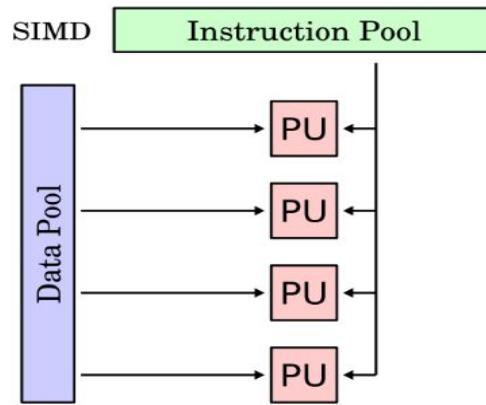
MISD computers can be useful in applications of a specialized nature:

- Robot vision
- When fault tolerance is required (military or aerospace application)
- Data can be processed by multiple machine and decision can be made on a majority principle



Single-Instruction, Multiple-DATA

- Multiple processing elements
- Each one works on its own data ,the physical memory might be shared
- All execute the same instruction at the same time

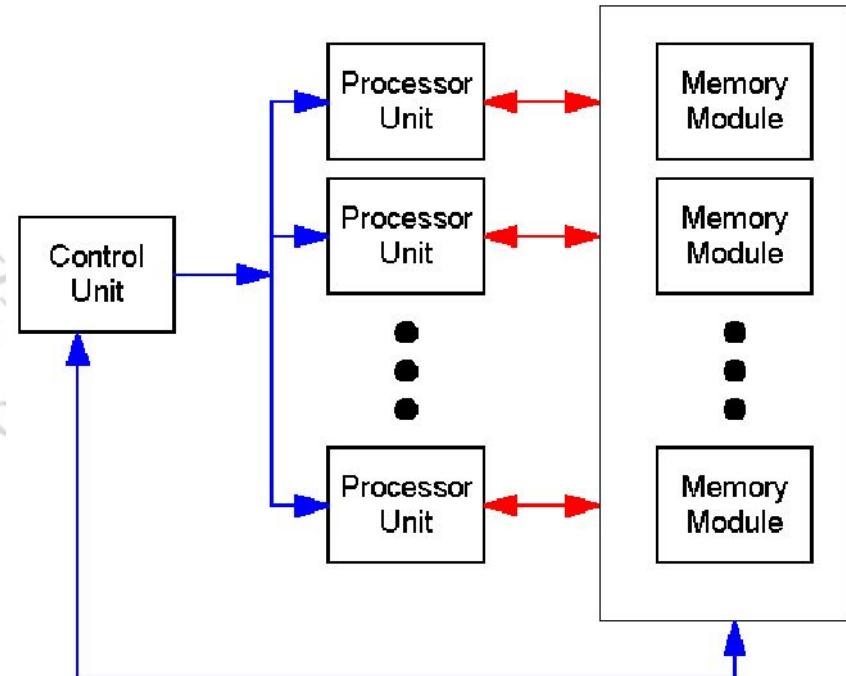


Single Instruction, Multiple Data Stream - SIMD

- Each processing element has an associated data memory
- Each instruction is executed on a different set of data by the different processors

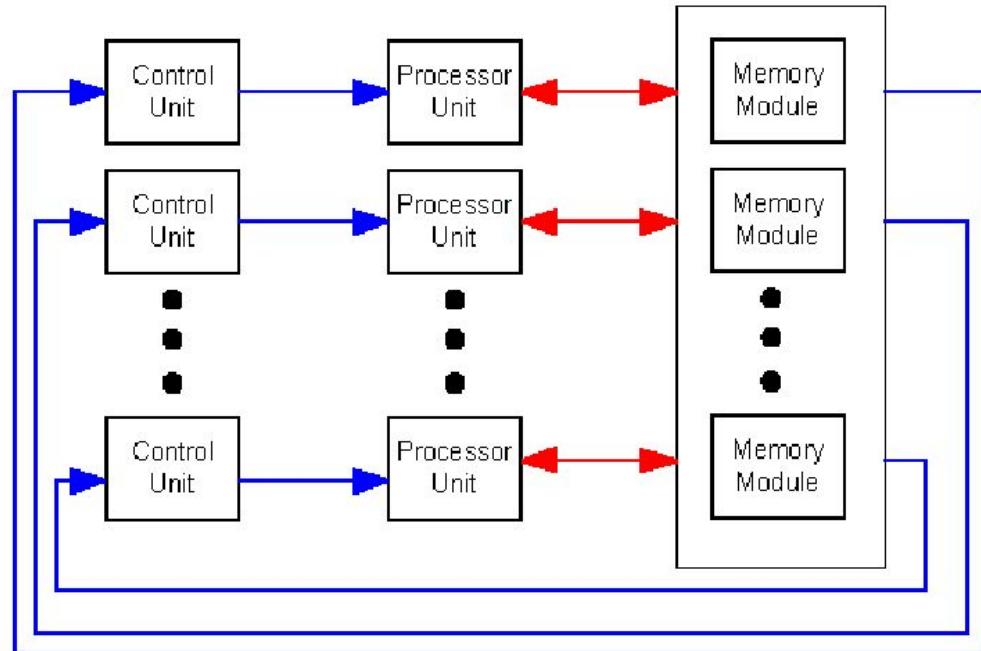
Applications:

- Image processing
- Matrix manipulations



Multiple Instruction, Multiple Data Stream- MIMD

- A set of processors simultaneously execute different instruction sequence on different data sets.
- Each processor can process all instructions necessary.

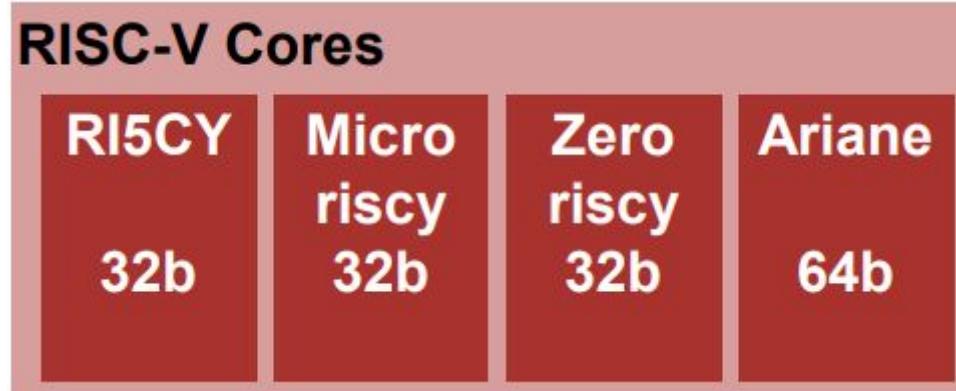


Feature	SISD	MISD	SIMD	MIMD
Instruction Stream	Single	Multiple	Single	Multiple
Data Stream	Single	Single	Multiple	Multiple
Type of Parallelism	sequential execution	Fault-tolerant	redundant	Data-level parallelism Task-level parallelism
Processing Units	Single	Multiple	Multiple	Multiple
Synchronization	Not required	High	Moderate	High
Fault Tolerance	Low	Very High	Low to Moderate	Moderate
Use Case Example	microcontrollers; legacy CPUs	Avionics; spacecraft systems;	GPUs; DSPs; image/video processing	Multicore CPUs; Servers; distributed systems
Hardware Cost	Low	Very High	Moderate	High
Scalability	Not scalable	Poor	Good	Very Good
Performance for Parallel Tasks	Poor	Poor to Moderate	High (data-parallel workloads)	Very High (mixed/ complex workloads)

Parallel Ultra Low Power (PULP)

- Project started in 2013
- A collaboration between University of Bologna and ETH Zürich
- Large team. In total we are about 60 people, not all are working on PULP
- Key goal is
- We were able to start with a clean slate, no need to remain compatible to legacy systems.
- How to get the most BANG for the ENERGY consumed in a computing system

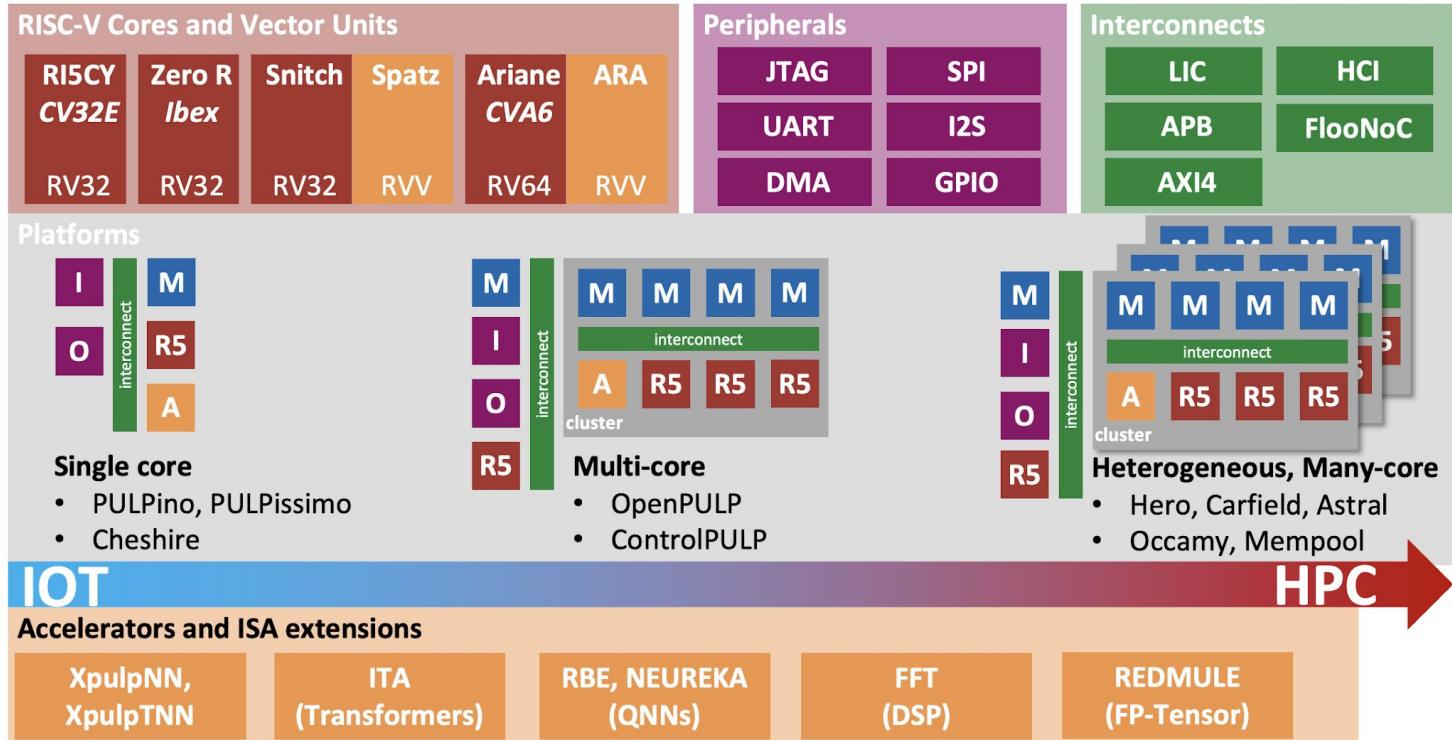
Development of several optimized RISC-V cores



Processing cores led to requirement of Peripherals

RISC-V Cores				Peripherals		Interconnect
RI5CY 32b	Micro riscy 32b	Zero riscy 32b	Ariane 64b	JTAG	SPI	Logarithmic interconnect
				UART	I2S	APB – Peripheral Bus
				DMA	GPIO	AXI4 – Interconnect

PULP RISC-V cores



32 bit			64 bit
Low Cost Core	Core with DSP enhancements	Floating-point capable Core	Linux capable Core
<ul style="list-style-type: none"> ■ Zero-riscy ■ RV32-ICM ■ Micro-riscy ■ RV32-CE <p>ARM Cortex-M0+</p>	<ul style="list-style-type: none"> ■ RI5CY ■ RV32-ICMX <ul style="list-style-type: none"> ■ SIMD ■ HW loops ■ Bit manipulation ■ Fixed point <p>ARM Cortex-M4</p>	<ul style="list-style-type: none"> ■ RI5CY+FPU ■ RV32-ICMFX <p>ARM Cortex-M4F</p>	<ul style="list-style-type: none"> ■ Ariane ■ RV64-IMAFDCX ■ Full privilege specification <p>ARM Cortex-A55</p>

Core

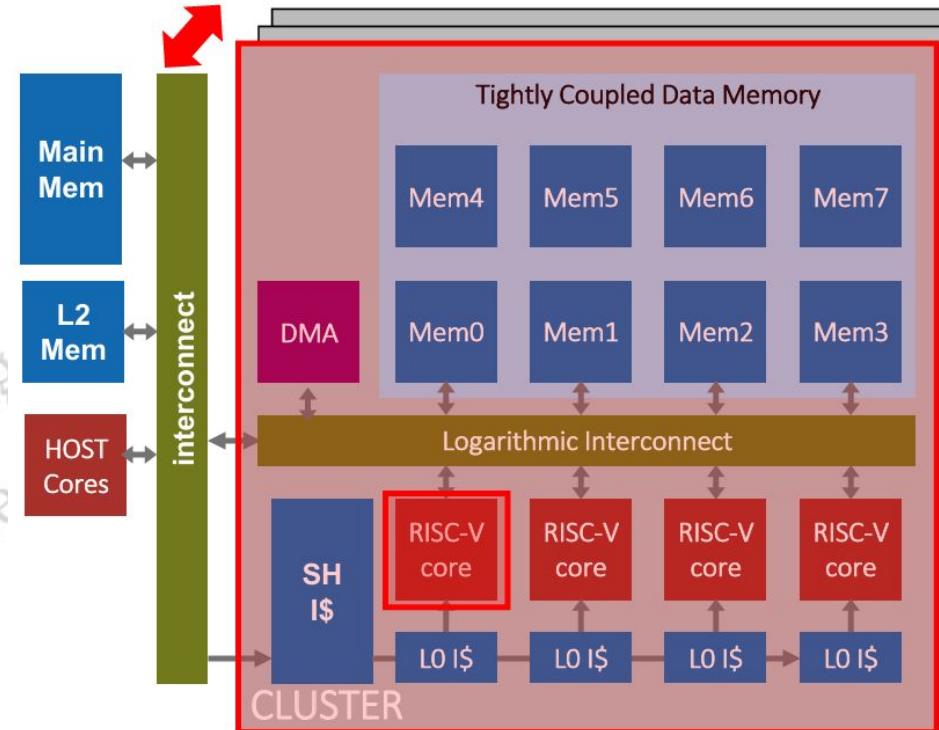
- Improving core efficiency with ISA and uAch extensions

Cluster

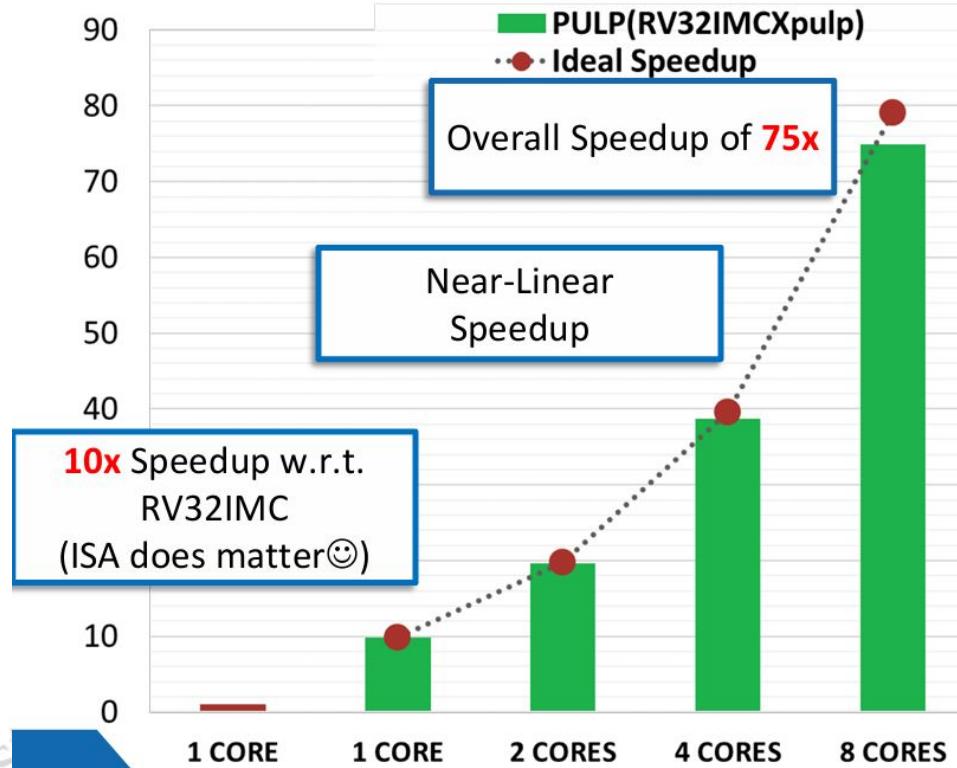
- Efficient shared-mem cluster
- From a few to thousand processing element

Full platform

- Heterogeneity: host, processor, accelerator
- One or multiple accelerators
- From chips to chiplets (2D to 3D)

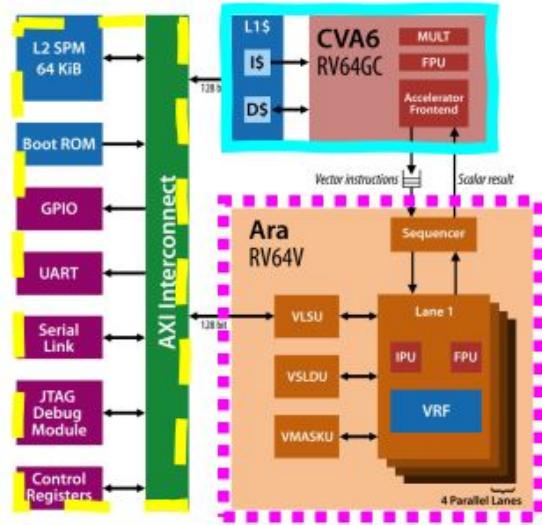


Performance

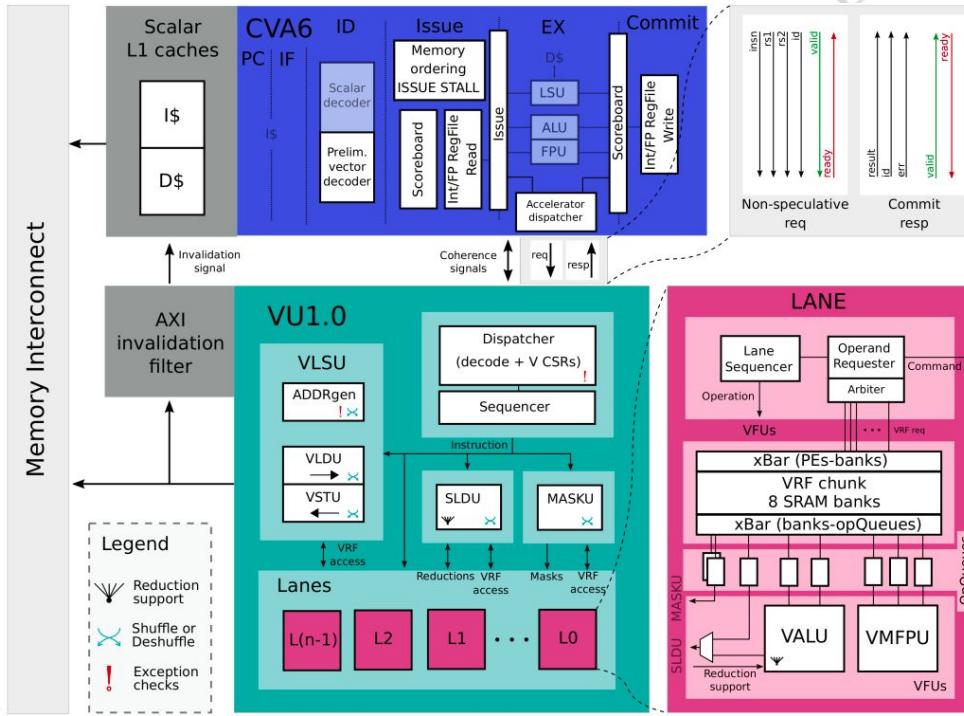


Introduction

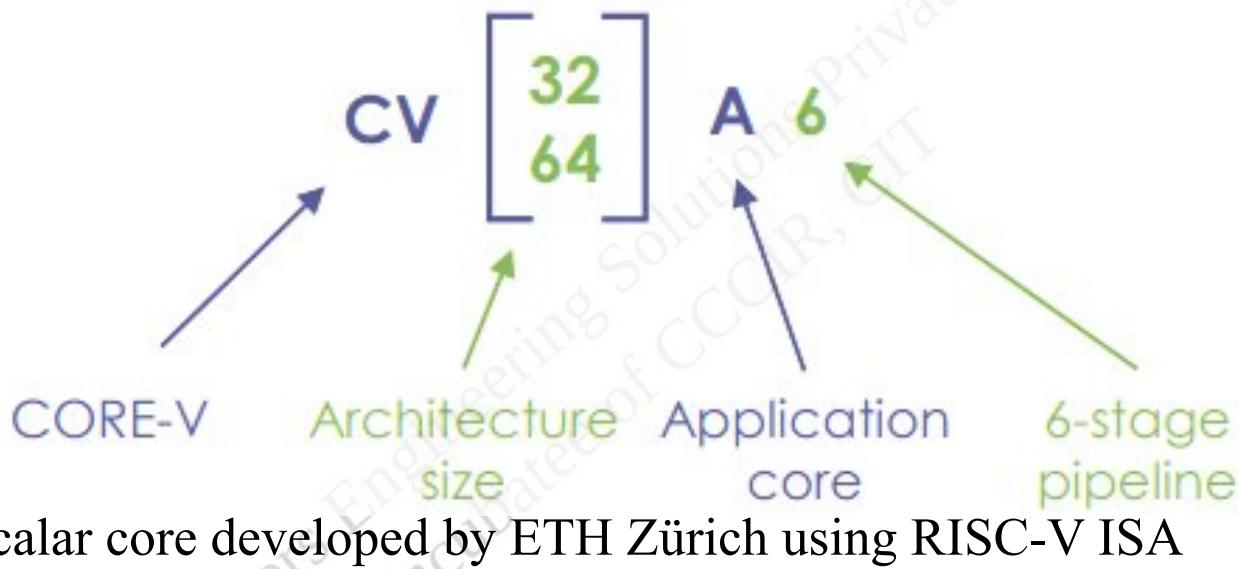
The PULP Ara is a 64-bit Vector Unit which is compatible with the RISC-V Vector extension version 1.0, working as a coprocessor to CORE-V's CVA6



PULP Ara Architecture

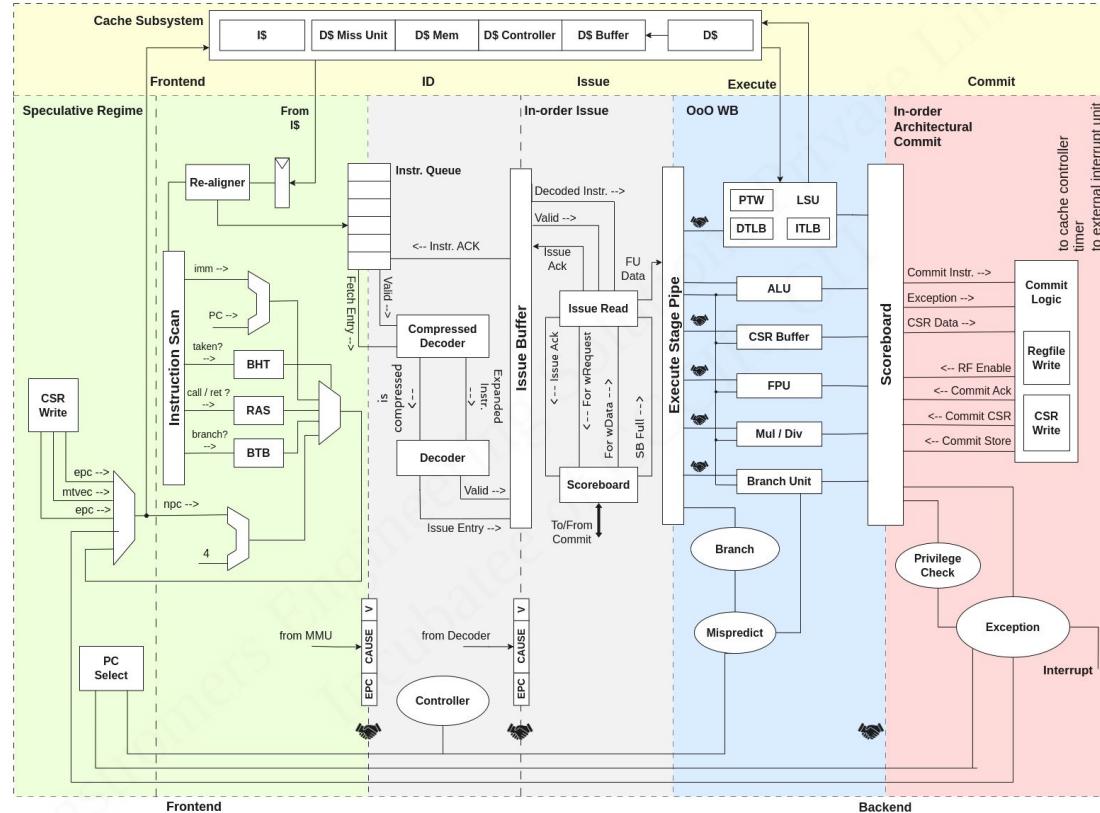


Open-source RISC-V application core

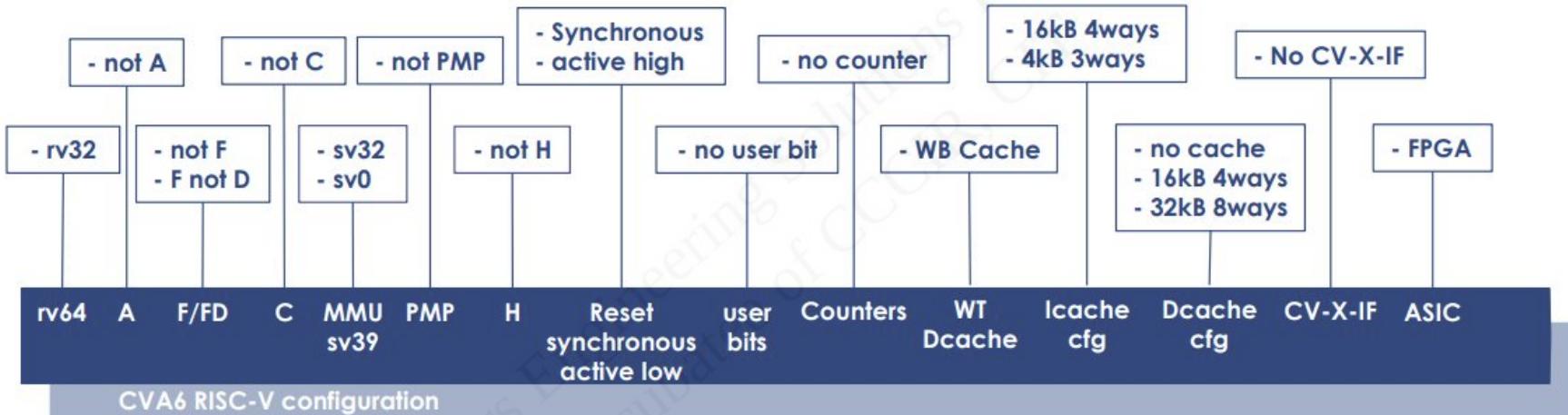


ARIANE Scalar core developed by ETH Zürich using RISC-V ISA

CVA6



CVA6: a highly configurable core



CVA6 core Architecture

- Fully compatible with the RISC-V open ISA
- 6-stage pipeline, single issue, branch prediction
- Soon dual-issue
- L1 data and instruction caches
- AXI4 interfaces M/S/U/H privileges
- Safe and secure features
- More upcoming (SECDED, DCLS)
- MMU and memory protection
- Ready for multi/many-core CPUs

An extendable core

- CV-X-IF interface to extend the CVA6 instruction set
- Custom extensions (cryptography, signal processing...)
- CV-X-IF specified by OpenHW Group Compiler support
- Seamless for supported standard extensions (e.g. B)
- LLVM should ease the support of custom extensions
- Inline ASM for specific processing

Software Ecosystem

Linux supported

- Available in 32 & 64 bit
- U-Boot, OpenSBI, BuildRoot Yocto
- FreeRTOS 32 & 64 bit supported
- GCC CVA6 features RISC-V extensions
- LLVM and custom extension support
- Debug: GDB, OpenOCD, Eclipse IDE



An academic project turning into an industrial-grade CPU core

- ARIANE donated to Open_HW Group by ETH Zürich



RISC-V International ("the Foundation")

- Specifies the open **RISC-V instruction set**
 - ✓ Simple & modular
 - ✓ 32- or 64-bit
 - ✓ Custom extensions
 - ✓ Covers a wide range of needs, **from MCU to HPC**
- Currently specifying **upcoming optional extensions**
- Hosts several **special interest groups (SIG)**
- Does not deliver implementations

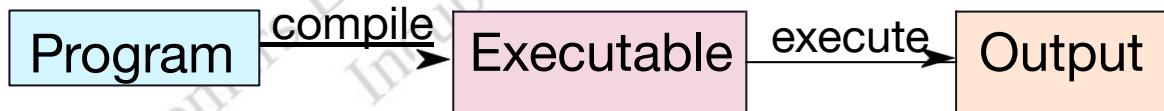


OpenHW Group

- **Not-for-profit corporation** steered by its members
- Goal: deliver **open-source IP for production SoCs**
 - ✓ RISC-V compatible cores
 - ✓ SoC IP blocks
 - ✓ Verification environment
 - ✓ Supporting SW and tools
- Permissive, open-source, export-friendly license

RISC-V Instructions

- One operation per instruction, at most one instruction per line
- Assembly instructions are related to C operations ($=$, $+$, $-$, $*$, $/$, $\&$, $|$, etc.)
 - Must be, since C code decomposes into assembly!
 - A single line of C may break up into several lines of RISC-V
- In Assembly Language, registers have no type, simply stores 0s and 1s; operation determines how register contents are treated



Assembly Instructions

- In assembly language, each statement (called an Instruction), executes exactly one of a short list of simple commands
- Unlike in C (and most other High Level Languages), each line of assembly code contains at most 1 instruction.
- Instructions are related to operations ($=$, $+$, $-$, $*$, $/$) in C

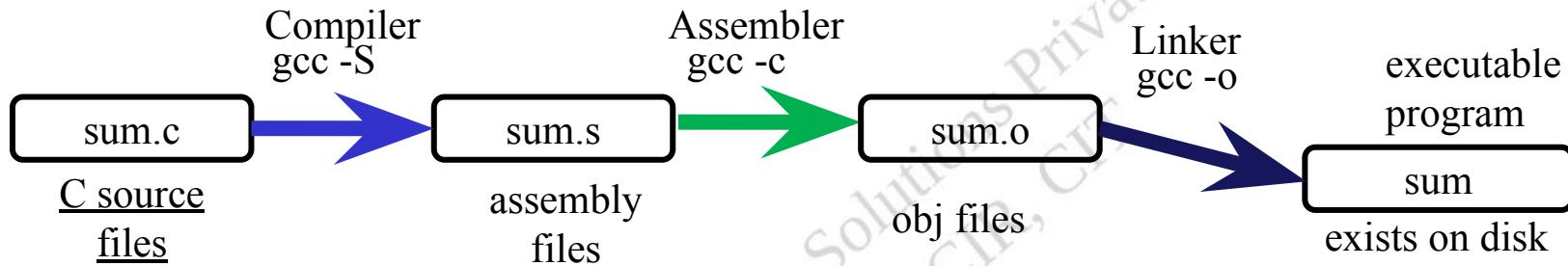
RISC-V Instructions

- Instruction Syntax is rigid:

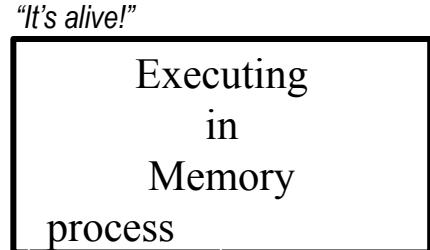
op dst, src1, src2

- 1 operator, 3 operands
 - op = operation name (“operator”)
 - dst = register getting result (“destination”)
 - src1 = first register for operation (“source 1”)
 - src2 = second register for operation (“source 2”)
 - Keep hardware simple via regularity

Software Design Flow



*When most people say “compile” they mean
the entire process:
*compile + assemble + link**



C variables vs. registers

- ❖ In C (and most High Level Languages) variables declared first and given a type.
E.g., int fahr, celsius;
char a, b, c, d, e;
- ❖ Each variable can ONLY represent a value of the type it was declared as (cannot mix and match int and char variables).
- ❖ In Assembly Language, the registers have no data type
 - Operation determines how register contents are treated

C:

```
car *c = malloc(sizeof(car));  
c->miles = 100;  
float mpg = get_mpg(c);  
free(c);
```

Assembly
language:

```
get_mpg(car*):  
    lw    a5,0(a0)  
    lw    a4,4(a0)  
    divw a5,a5,a4  
    fcvt.s.w      fa0,a5  
    ret
```

Machine code:

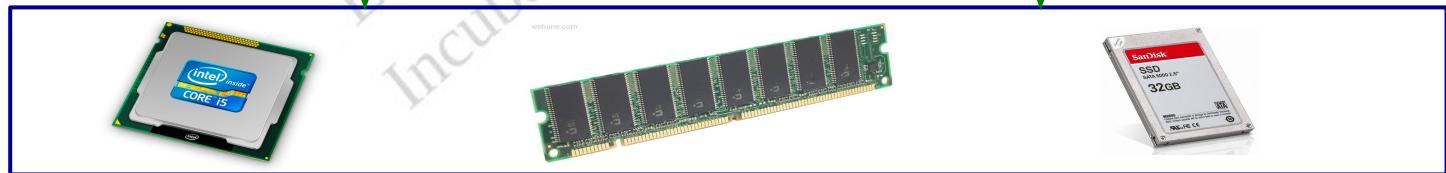
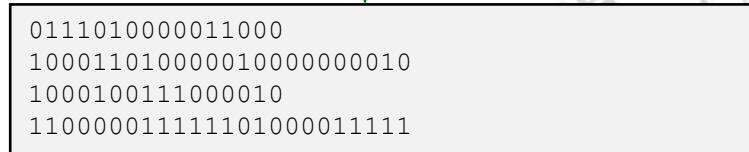
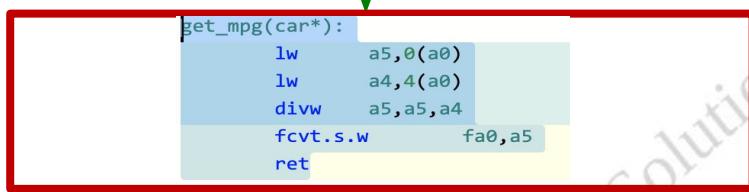
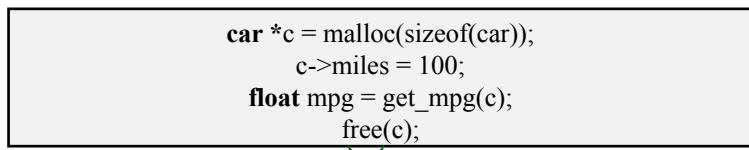
```
0111010000011000  
100011010000010000000010  
1000100111000010  
110000011111101000011111
```

Computer
system:



Memory & data
Arrays & structs
Integers & floats
RISC V assembly
Procedures & stacks
Executables
Memory & caches
Processor Pipeline
Performance
Parallelism

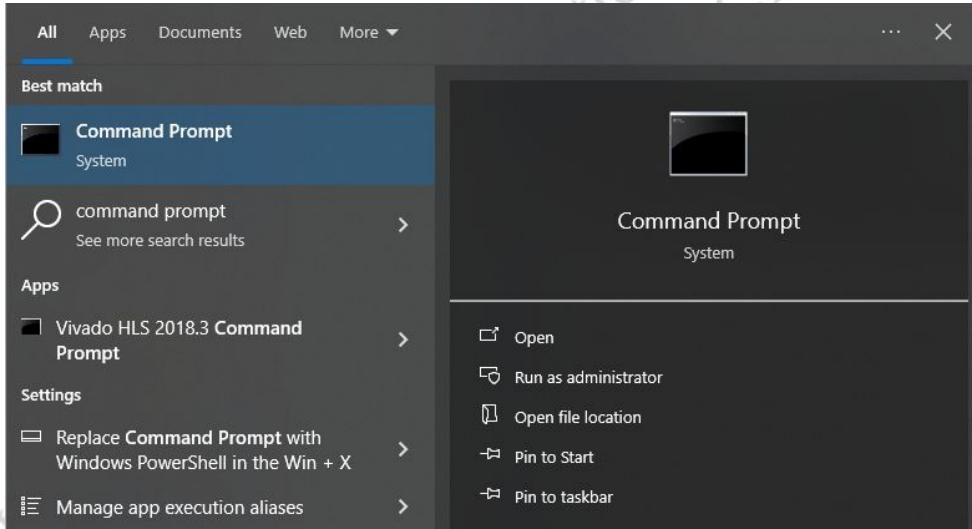
OS:



RISC-V Programming

Launch the WSL software

- Search for command prompt in windows and type the following command



Initial Configuration

- Launch command prompt using **wsl -d Ubuntu**
- To change from existing windows directory **cd /** to the main working directory in Ubuntu

```
root@DESKTOP-U5VUKL1:/mnt/c/Users/Copi-004
Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Copi-004>wsl -d Ubuntu
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Fri Feb 28 16:35:24 IST 2025

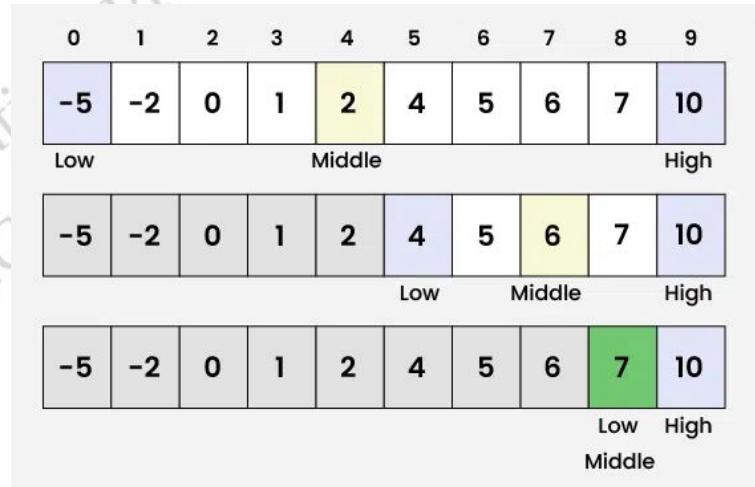
System load:  0.52      Processes:          9
Usage of /home: unknown  Users logged in:   0
Memory usage: 17%       IPv4 address for eth0: 192.168.0.138
Swap usage:   0%

=> /mnt/f is using 94.0% of 146.48GB

This message is shown once a day. To disable it please create the
/root/.hushlogin file.
root@DESKTOP-U5VUKL1:/mnt/c/Users/Copi-004# cd /
```

Binary Search

- Binary search is used to find a target in a sorted array.
- It works by repeatedly halving the search range.
- The middle element is compared with the target.
- If the middle element matches the target, the search stops.
- If the target is smaller, search continues in the left half.



Code Compilation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = binary_search

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name
```

- To ~~copy the code from the internet and type main.c~~ follow the steps given below followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c  
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../../  
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Code Execution

```
copi001-OptiPlex-7050:~/ara$ make -C apps bin/binary_search
Building directory '/home/copi-001/ara/apps'
binary_search && if [ -d script ]; then python3 script/gen_data.py
/home/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh_zvfm
scalable-vectorization=on -mllvm -riscv-v-vector-bits-min
me/copi-001/ara/apps/common -std=gnu99 -O3 -ffast-math -fno-optimizations
-command-line-argument -ffunction-sections -fdata-sections
/home/copi-001/ara/apps/common/script/align_sections.sh
me/copi-001/ara/apps/common/link.ld && cp /home/copi-001/ara/
i-001/ara/apps/common/script/align_sections.sh 4 /home/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh_zvfm
scalable-vectorization=on -mllvm -riscv-v-vector-bits-min
me/copi-001/ara/apps/common -std=gnu99 -O3 -ffast-math -fno-optimizations
-command-line-argument -ffunction-sections -fdata-sections
i-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh_zvfm
scalable-vectorization=on -mllvm -riscv-v-vector-bits-min
me/copi-001/ara/apps/common -std=gnu99 -O3 -ffast-math -fno-optimizations
```

```
#include <stdio.h>
#include <printf.h>
int binarySearch(int array[], int x, int low, int high) {
    // Repeat until the pointers low and high meet each other
    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (array[mid] == x)
            return mid;

        if (array[mid] < x)
            low = mid + 1;

        else
            high = mid - 1;
    }
    return -1;
}

int main(void) {
    int array[] = {3, 4, 5, 6, 7, 8, 9};
    int n = sizeof(array) / sizeof(array[0]);
    int x = 4;
    int result = binarySearch(array, x, 0, n - 1);
    if (result == -1)
        printf("Not found");
    else
        printf("Element is found at index %d\n", result);
    return 0;
}
```

Code Execution

```
#include <stdio.h>
#include <printf.h>
int binarySearch(int array[], int x,
int low, int high) {
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (array[mid] == x)
            return mid;
        if (array[mid] < x)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
int main(void) {
    int array[] = {3, 4, 5, 6, 7, 8, 9};
    int n = sizeof(array) / sizeof(array[0]);
    int x = 4;
    int result = binarySearch(array, x, 0, n - 1);
    if (result == -1)
        //printf("Not found");
    // else
        printf("Element is found at index %d\n", result);
    return 0;
}
```

Simulation on ARA

For simulation command : make -C hardware simv app=folder_name

Where, folder_name is the program to be executed.

folder_name = binary_search

```
~/ara$ make -C hardware simv app=folder_name
```

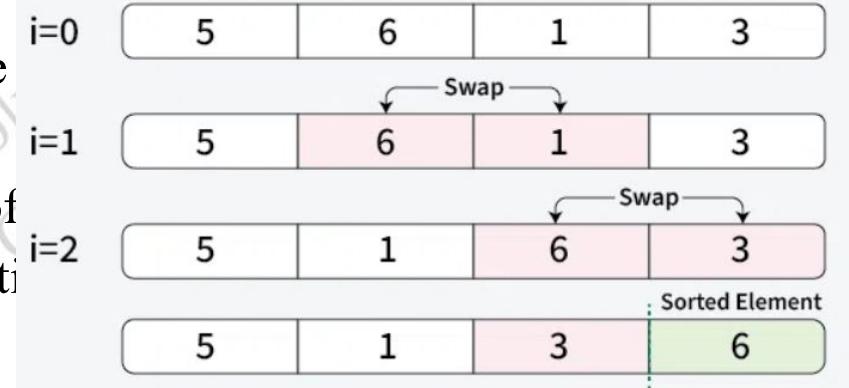
```
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process:
$ kill -USR1 29e0

Simulation running, end by pressing CTRL-c.
Element is found at index 1
[hw-cycles]:          0
[5298] -Info: ara_tb_verilator.sv:49: Assertion failed in TOP.
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv:52: Veril
Received $finish() from Verilog, shutting down simulation.

Simulation statistics
=====
Executed cycles: a59
Wallclock time:  1.104 s
Simulation speed: 2399.46 cycles/s (2.39946 kHz)
```

Bubble Sort

- Bubble Sort is used, to repeatedly swaps adjacent elements if they are in the wrong order.
- It sorts in multiple passes, moving the position in each pass.
- In each pass, only the unsorted portion of
- Elements are compared and swapped until



Code Compilation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = bubble_sort

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c  
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../../  
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Code Execution

```
#include <stdio.h>
#include <printf.h>
void swap(int* arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(arr, j, j + 1);
}
```

```
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    int arr[] = { 5, 1, 4, 2, 8 };
    int N = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, N);
    printf("Sorted array: ");
    printArray(arr, N);
    return 0;
}
```

Code Execution

```
~/ara$ make -C apps bin/bubble_sort
[~/copi-001/ara/apps'
ot ]; then python3 script/gen_data.py > data.S ; else touch data.S; fi
tv-llvm/bin/clang -march=rv64gcv_zfh_zvfh0p1 -menable-experimental-extension
tion=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -target-feature -Xclar
common -std=gnu99 -O3 -ffast-math -fno-common -fno-built-in printf -DNR_LAN
ument -ffunction-sections -fdata-sections -c bubble_sort/data.S -o bubble_
s/common/script/align_sections.sh
common/link.ld && cp /home/copi-001/ara/apps/common/arch.link.ld /home/copi-
script/align_sections.sh 4 /home/copi-001/ara/apps/common/link.ld
tv-llvm/bin/clang -march=rv64gcv_zfh_zvfh0p1 -menable-experimental-extension
tion=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -target-feature -Xclar
common -std=gnu99 -O3 -ffast-math -fno-common -fno-built-in printf -DNR_LAN
ument -ffunction-sections -fdata-sections -c bubble_sort/main.c -o bubble_
tv-llvm/bin/clang -march=rv64gcv_zfh_zvfh0p1 -menable-experimental-extension
tion=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -target-feature -Xclar
common -std=gnu99 -O3 -ffast-math -fno-common -fno-built-in printf -DNR_LAN
ument -ffunction-sections -fdata-sections -c common/crt0.S -o common/crt0_
tv-llvm/bin/clang -march=rv64gcv_zfh_zvfh0p1 -menable-experimental-extension
tion=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -target-feature -Xclar
```

```
#include <stdio.h>
#include <printf.h>
// Swap function
void swap(int* arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
        // Last i elements are already
        // in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(arr, j, j + 1);
}
```

Simulation on ARA

For simulation command : make -C hardware simv app=folder_name

Where, folder_name is the program to be executed.

folder_name = bubble_sort

```
~/ara$ make -C hardware simv app=folder_name
```

```
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 3a9a

Simulation running, end by pressing CTRL-C.
Sorted array: 1 2 4 5 8
[hw-cycles]: 0
[9230] -Info: ara_tb_verilator.sv:49: Assertion failed
- /home/copi-001/ara/hardware/tb/ara_tb_verilator.sv:
Received $finish() from Verilog, shutting down simulation

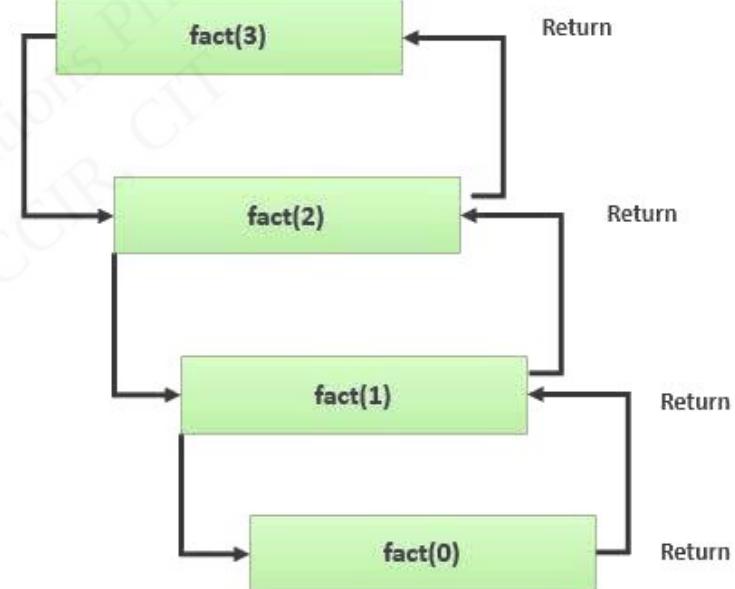
Simulation statistics
=====
Executed cycles: 1207
Wallclock time: 1.997 s
Simulation speed: 2310.97 cycles/s (2.31097 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
```

Factorial

The factorial of a number, denoted as ' $n!$ ', is the product of all positive integers up to n .

Essential in permutations, combinations, and probability.

It simplifies complex calculations, like determining possible seating arrangements



Code Compilation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored. Example :
folder_name = Program_name = factorial

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

- To create a new file to write the C code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c  
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../../  
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Code Execution

```
#include<stdio.h>
#include <printf.h>

unsigned int factorial(unsigned int N)
{
    int fact = 1,i;

    for(i =1; i <= N; i++)
    {
        fact *= i;
    }
    return fact;
}
```

```
int main()
{
    int N = 5;
    int fact = factorial(N);
    printf("Factorial of %d is %d", N, fact);
    return 0;
}
```

Code Execution

```
l@copi001-OptiPlex-7050:~$ cd ara
l@copi001-OptiPlex-7050:~/ara$ make -C apps bin/factorial
Entering directory '/home/copi-001/ara/apps'
factorial && if [ -d script ]; then python3 script/gen_data.py > data.S; fi
/home/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh0p1
experimental-extensions -mabi=lp64d -mno-relax -fuse-lld=lld -fvectoriz
able-vectorization=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang
-include -Xclang +no-optimized-zero-stride-load -mcmodel=medany -I/home/
apps/common -std=gnu99 -O3 -ffast-math -fno-common -fno-builtin-prin
t -DVLEN=4096 -Wunused-variable -Wall -Wextra -Wno-unused-command-line-
option -ffunction-sections -fdata-sections -c factorial/data.S -o factorial.o
o
x /home/copi-001/ara/apps/common/script/align_sections.sh
/home/copi-001/ara/apps/common/link.ld && cp /home/copi-001/ara/apps/
link.ld /home/copi-001/ara/apps/common/link.ld
/home/copi-001/ara/apps/common/script/align_sections.sh 4 /home/copi-001/a
n/link.ld
/home/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh0p1
experimental-extensions -mabi=lp64d -mno-relax -fuse-lld=lld -fvectoriz
able-vectorization=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang
-include -Xclang +no-optimized-zero-stride-load -mcmodel=medany -I/home/
apps/common -std=gnu99 -O3 -ffast-math -fno-common -fno-builtin-prin
t -DVLEN=4096 -Wunused-variable -Wall -Wextra -Wno-unused-command-line-
option -ffunction-sections -fdata-sections -c factorial/data.S -o factorial.o
o
```

```
#include<stdio.h>
#include <printf.h>

unsigned int factorial(unsigned int N)
{
    int fact = 1,i;

    for(i =1; i <= N; i++)
    {
        fact *= i;
    }
    return fact;
}

int main()
{
    int N = 5;
    int fact = factorial(N);
    printf("Factorial of %d is %d", N, fact);
    return 0;
}
```

Simulation on ARA

For simulation command : make -C hardware simv app=folder_name

Where, folder_name is the program.

folder_name = factorial

```
ara$ make -C hardware simv app=factorial
```

```
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 2208

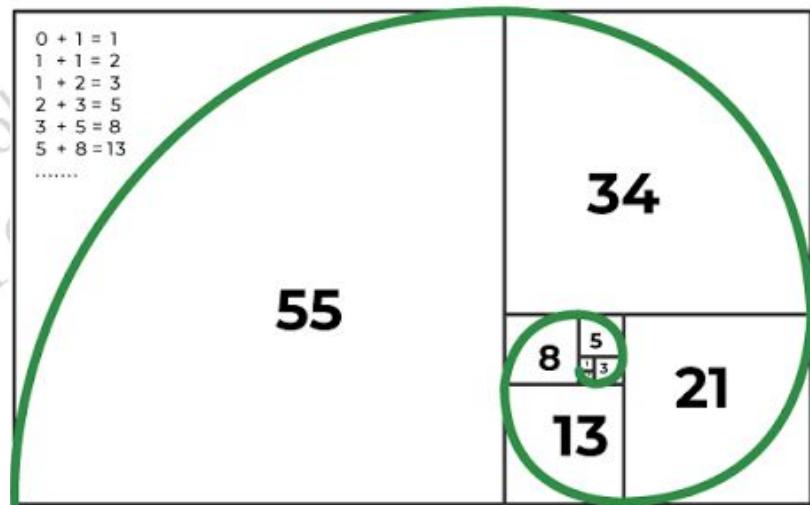
Simulation running, end by pressing CTRL-c.
Factorial of 5 is 120[hw-cycles]:          0
[5616] -Info: ara_tb_verilator.sv:49: Assertion failed
Core Test *** SUCCESS *** (tohost = 0)
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv
Received $finish() from Verilog, shutting down simulation

Simulation statistics
=====
Executed cycles: af8
Wallclock time:  1.16 s
Simulation speed: 2420.69 cycles/s (2.42069 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
```

Fibonacci Series

The Fibonacci series is the sequence where each number is the sum of the previous two numbers of the sequence.

The first two numbers are 0 and 1 which are used to generate the whole series.



Code Compilation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored. Example :
folder_name = Program_name = fibonacci

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

- To create a new file to write the C code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c  
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../../  
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Code Execution

```
#include <printf.h>
#include <stdio.h>
int main() {
    int i, n;
    // initialize first and second terms
    int t1 = 0, t2 = 1;
    // initialize the next term (3rd term)
    int nextTerm = t1 + t2;
    // print the first two terms t1 and t2
    printf("Fibonacci Series: %d, %d, ", t1, t2);
    // print 3rd to nth terms
    for (i = 3; i <= n; ++i) {
        printf("%d, ", nextTerm);
        t1 = t2;
        t2 = nextTerm;
        nextTerm = t1 + t2;
    }
    return 0;
}
```

Code Execution

```
: Leaving directory '/home/copi-001/ara/hardware'  
-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/fibonacci  
: Entering directory '/home/copi-001/ara/apps'  
fibonacci && if [ -d script ]; then python3 script/gen_data.py > data.S ; el  
ouch data.S; fi  
e/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh_zvh0p1 -mena  
experimental-extensions -mabi=lp64d -mno-relax -fuse-ld=lld -fvectorize -mll  
scalable-vectorization=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -targ  
eature -Xclang +no-optimized-zero-stride-load -mcmodel=medany -I/home/copi-0  
ra/apps/common -std=gnu99 -O3 -ffast-math -fno-common -fno-builtin-printf -  
LANES=4 -DVLEN=4096 -Wunused-variable -Wall -Wextra -Wno-unused-command-line  
ument -ffunction-sections -fdata-sections -c fibonacci/data.S -o fibonacci/  
.S.o  
d +x /home/copi-001/ara/apps/common/script/align_sections.sh  
f /home/copi-001/ara/apps/common/link.ld && cp /home/copi-001/ara/apps/commo  
n/link.ld /home/copi-001/ara/bin/fibonacci/link.ld
```

```
#include <printf.h>  
#include <stdio.h>  
int main() {  
  
    int i, n;  
  
    // initialize first and second terms  
    int t1 = 0, t2 = 1;  
  
    // initialize the next term (3rd term)  
    int nextTerm = t1 + t2;  
  
    // print the first two terms t1 and t2  
    printf("Fibonacci Series: %d, %d, ", t1, t2);  
  
    // print 3rd to nth terms  
    for (i = 3; i <= n; ++i) {  
        printf("%d, ", nextTerm);  
        t1 = t2;  
        t2 = nextTerm;  
        nextTerm = t1 + t2;  
    }  
  
    return 0;  
}
```

Simulation on ARA

For simulation command : make -C hardware simv app=folder_name

Where, folder_name is the program.

folder_name = fibonacci

```
ara$ make -C hardware simv app=factorial
```

```
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 2208

Simulation running, end by pressing CTRL-c.
Factorial of 5 is 120[hw-cycles]:          0
[5616] -Info: ara_tb_verilator.sv:49: Assertion failed
Core Test *** SUCCESS *** (tohost = 0)
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv
Received $finish() from Verilog, shutting down simulation

Simulation statistics
=====
Executed cycles: af8
Wallclock time:  1.16 s
Simulation speed: 2420.69 cycles/s (2.42069 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
```

Palindrome

An integer is a palindrome if the reverse of that number is equal to the original number.

A simple method is to first reverse all the digits of a given number using arithmetic operations.

Then compare the reverse of the number with a given number.

Palindrome

12321 **12321**
Reversing

Not Palindrome

1232 **2321**
Reversing

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = palindrome

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Code Execution

```
#include <stdio.h>
#include <printf.h>
int main() {
    int n = 121, reversed = 0, remainder, original;
    printf("Enter integer: %d\n", n);
    original = n;
    // reversed integer is stored in reversed
    variable
    while (n != 0) {
        remainder = n % 10;
        reversed = reversed * 10 + remainder;
        n /= 10;
    }
    // palindrome if original and reversed are equal
    if (original == reversed)
        printf("%d is a palindrome.\n", original);
    else
        printf("%d is not a palindrome.", original);
    return 0;
}
```

Code Execution

```
File Edit View Search Terminal Help
copi-001@copi001-OptiPlex-7050:~$ cd ara
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/palindrome
make: Entering directory '/home/copi-001/ara/apps'
cd palindrome && if [ -d script ]; then python3 script/gen_data.py > data.S ; else touch data.S; fi
/home/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh_zvh0p1 -menable-experimental-extensions -mabi=lp64d -mno-relax -fuse-lld=lld -fvectorize -mllvm -scalable-vectorization=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -target-feature -Xclang +no-optimized-zero-stride-load -mcmodel=medany -I/home/copi-001/ara/apps/common -std=gnu99 -O3 -ffast-math -fno-common -fno-builtin-printf -DNR_LANES=4 -DVLEN=4096 -Wunused-variable -Wall -Wextra -Wno-unused-command-line-argument -ffunction-sections -fdata-sections -c palindrome/data.S -o palindrome/data.S.o
chmod +x /home/copi-001/ara/apps/common/script/align_sections.sh
rm -f /home/copi-001/ara/apps/common/link.ld && cp /home/copi-001/ara/apps/common/arch.link.ld /home/copi-001/ara/apps/common/link.ld
/home/copi-001/ara/apps/common/script/align_sections.sh 4 /home/copi-001/ara/apps/common/link.ld
/home/copi-001/ara/install/riscv-llvm/bin/clang -march=rv64gcv_zfh_zvh0p1 -menable-experimental-extensions -mabi=lp64d -mno-relax -fuse-lld=lld -fvectorize -mllvm -scalable-vectorization=on -mllvm -riscv-v-vector-bits-min=2048 -Xclang -target-feature -Xclang +no-optimized-zero-stride-load -mcmodel=medany -I/home/copi-001/ara/apps/common -std=gnu99 -O3 -ffast-math -fno-common -fno-builtin-printf -
```

```
#include <stdio.h>
int main() {
    int n, reversed = 0, remainder, original;
    printf("Enter an integer: ");
    scanf("%d", &n);
    original = n;

    // reversed integer is stored in reversed variable
    while (n != 0) {
        remainder = n % 10;
        reversed = reversed * 10 + remainder;
        n /= 10;
    }

    // palindrome if original and reversed are equal
    if (original == reversed)
        printf("%d is a palindrome.\n", original);
    else
        printf("%d is not a palindrome.", original);

    return 0;
}
```

Simulation on ARA

For simulation command :

```
make -C hardware simv app=folder_name
```

Where, folder_name is the program.

folder_name = palindrome

```
~/ara$ make -C hardware simv app=palindrome
```

```
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 12c2

Simulation running, end by pressing CTRL-c.
Enter integer: 121
121 is a palindrome.

[hw-cycles]:          0
[7002] -Info: ara_tb_verilator.sv:49: Assertion failed
Core Test *** SUCCESS *** (tohost = 0)
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv:49
Received $finish() from Verilog, shutting down simulation.

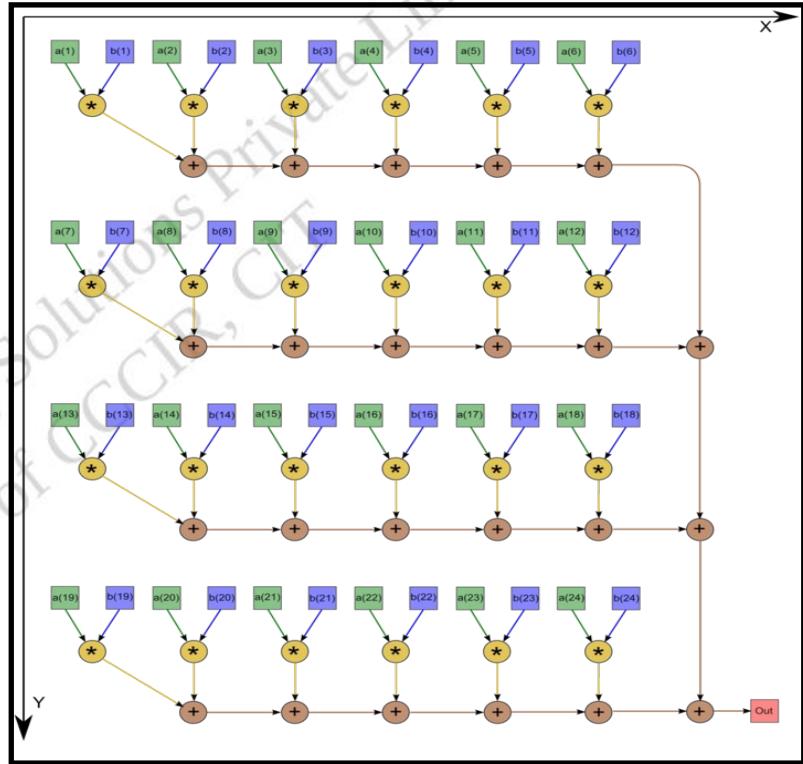
Simulation statistics
=====
Executed cycles: 121
Wallclock time:  1.524 s
Simulation speed: 2297.24 cycles/s (2.29724 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
```

Dot Product

A dot product is a combination of multiplication and addition between two vectors. Given two vectors

$$A = [aa\ bb\ cc] \text{ and } D = [dd\ ee\ ff]$$

The dot product $A \cdot D$ is defined by ,
 $A \cdot D = ad + be + cf$



Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = dot_product

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Dot product

```
#include <stdio.h>

int main()
{
    int a[3] = {1, 2, 3};
    int b[3] = {4, 5, 6};
    int sum = 0;
    for (int i = 0; i < 3; ++i)
    {
        sum += a[i] * b[i];
    }
    printf("The inner product of two vectors is %d", sum);
    return 0;
}
```

Simulation on ARA

For simulation command :

```
make -C hardware simv app=folder_name
```

Where, folder_name is the program.

folder_name = dot_product

```
[new offset: 0x0 in file: /home/copi-001/ara/hardware/tb/ara_tb_verilator.sv]
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 12c2
Simulation running, end by pressing CTRL-c.
Enter integer: 121
121 is a palindrome.
[hw-cycles]: 0
[7002] -Info: ara_tb_verilator.sv:49: Assertion failed
Core Test *** SUCCESS *** (tohost = 0)
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv:49
Received $finish() from Verilog, shutting down simulation
Simulation statistics
=====
Executed cycles: 121
Wallclock time: 1.524 s
Simulation speed: 2297.24 cycles/s (2.29724 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
```

Length of last string

```
#include <stdio.h>
#include <string.h>
int main() {

    const char* str1 = "Hello World";
    const char* str2 = "Length of the last word is ";

    printf("%s\n", str1);
    int len = strlen(str1);
    int count = 0;
    int i = len - 1;

    while (i >= 0 && str1[i] == ' ') {
        i--;
    }

    while (i >= 0 && str1[i] != ' ') {
        count++;
        i--;
    }

    printf("%s%d\n", str2, count);

    return 0;
}
```

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : `folder_name = Program_name = string`

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Simulation on ARA

For simulation command :

```
make -C hardware simv app=folder_name
```

Where, folder_name is the program.

folder_name = string

```
[new offset: 0x0 in file: /home/copi-001/ara/hardware/tb/ara_tb_verilator.sv]
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 12c2
Simulation running, end by pressing CTRL-c.
Enter integer: 121
121 is a palindrome.
[hw-cycles]:          0
[7002] -Info: ara_tb_verilator.sv:49: Assertion failed
Core Test *** SUCCESS *** (tohost = 0)
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv
Received $finish() from Verilog, shutting down simulation

Simulation statistics
=====
Executed cycles: 121
Wallclock time:  1.524 s
Simulation speed: 2297.24 cycles/s (2.29724 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
```

Matrix Multiplication

```
#include <stdio.h>
#define N 3 // Matrix size (N x N)
void matrix_multiply(int A[N][N],
int B[N][N], int C[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            C[i][j] = 0;
            for (int k = 0; k < N; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

int main() {
    int A[N][N] = {{1, 2, 3},
                   {4, 5, 6},
                   {7, 8, 9}};
    int B[N][N] = {{9, 8, 7},
                   {6, 5, 4},
                   {3, 2, 1}};
    int C[N][N] = {0};
    matrix_multiply(A, B, C);
    printf("Resultant Matrix C:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = matrix

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Simulation on ARA

For simulation command :

make -C hardware simv app=folder_name

Where, folder_name is the program.

folder_name = matrix

```
[new offset: 0x0 in file: /home/copi-001/ara/hardware/tb/ara_tb_verilator.sv]
Simulation of Ara
=====
Tracing can be toggled by sending SIGUSR1 to this process
$ kill -USR1 12c2
Simulation running, end by pressing CTRL-c.
Enter integer: 121
121 is a palindrome.
[hw-cycles]:          0
[7002] -Info: ara_tb_verilator.sv:49: Assertion failed
Core Test *** SUCCESS *** (tohost = 0)
- ./home/copi-001/ara/hardware/tb/ara_tb_verilator.sv
Received $finish() from Verilog, shutting down simulation

Simulation statistics
=====
Executed cycles: 121
Wallclock time:  1.524 s
Simulation speed: 2297.24 cycles/s (2.29724 kHz)
make: Leaving directory '/home/copi-001/ara/hardware'
exit 2013-01-01 09:18:47.500 +0530
```

Matrix Transpose

```
#include <stdio.h>
#define N 4
void transpose(int A[][N], int B[][N])
{
    int i, j;
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            // Assigns the transpose of
element A[j][i] to
            // B[i][j]
            B[i][j] = A[j][i];
}
int main()
{
```

```
    int A[N][N] = { { 1, 1, 1, 1 },
                    { 2, 2, 2, 2 },
                    { 3, 3, 3, 3 },
                    { 4, 4, 4, 4 } };
    int B[N][N], i, j;
    transpose(A, B);
    printf("Result matrix is \n");
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++)
            printf("%d ", B[i][j]);
        printf("\n");
    }
    return 0;
}
```

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = transpose

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Matrix Addition

```
#include <stdio.h>
#define ROWS 2
#define COLUMNS 3
void addMatrices(int
A[ROWS][COLUMNS], int
B[ROWS][COLUMNS], int
C[ROWS][COLUMNS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
}
int main() {
    int A[ROWS][COLUMNS] = {
        {1, 2, 3},
        {4, 5, 6}
    };
    int B[ROWS][COLUMNS] = {
        {6, 5, 4},
        {3, 2, 1}
    };
    int C[ROWS][COLUMNS];
    addMatrices(A, B, C);
    printf("Resultant Matrix (A + B):\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = addition

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Matrix Subtraction

```
#include <stdio.h>
#define ROWS 2
#define COLUMNS 3
void subtractMatrices(int
A[ROWS][COLUMNS], int
B[ROWS][COLUMNS], int
C[ROWS][COLUMNS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            C[i][j] = A[i][j] - B[i][j];
        }
    }
}
int main() {
    int A[ROWS][COLUMNS] = {
        {10, 20, 30},
        {40, 50, 60}
    };
    int B[ROWS][COLUMNS] = {
        {1, 2, 3},
        {4, 5, 6}
    };
    int C[ROWS][COLUMNS];
    subtractMatrices(A, B, C);
    printf("Resultant Matrix (A - B):\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = subtraction

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```

Matrix Division

```
#include <stdio.h>
#define ROWS 2
#define COLUMNS 3
void divideMatrices(int A[ROWS][COLUMNS], int
B[ROWS][COLUMNS], float
C[ROWS][COLUMNS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            if (B[i][j] != 0)
                C[i][j] = (float)A[i][j] / B[i][j];
            else
                C[i][j] = 0.0; // or handle error
        }
    }
}
int main() {
    int A[ROWS][COLUMNS] = {
        {10, 20, 30},
        {40, 50, 60}
    };
    int B[ROWS][COLUMNS] = {
        {2, 4, 5},
        {8, 10, 15}
    };
    float C[ROWS][COLUMNS];
    divideMatrices(A, B, C);
    printf("Resultant Matrix (A / B):\n");
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLUMNS; j++) {
            printf("%.2f ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Source file creation

- To create a new folder in the directory type the command mkdir followed by the folder name, in which the saved program will be stored.
- Example : folder_name = Program_name = division

```
copi-001@copi001-OptiPlex-7050:~/ara$ cd apps  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ mkdir folder_name  
copi-001@copi001-OptiPlex-7050:~/ara/apps$ cd folder_name/
```

Code Compilation

- To create a new file to write the c code using editor type command nano followed by .c extension.

```
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ nano main.c
copi-001@copi001-OptiPlex-7050:~/ara/apps/folder_name$ cd ../..
copi-001@copi001-OptiPlex-7050:~/ara$ make -C apps bin/folder_name
```