

Special Function Units

Instructor: Siting Liu

School of Information Science and Technology (SIST), ShanghaiTech University

Shanghai Institute of Semiconductor Innovation (SISI)

<https://faculty.sist.shanghaitech.edu.cn/liust/EE219/EE219.html>

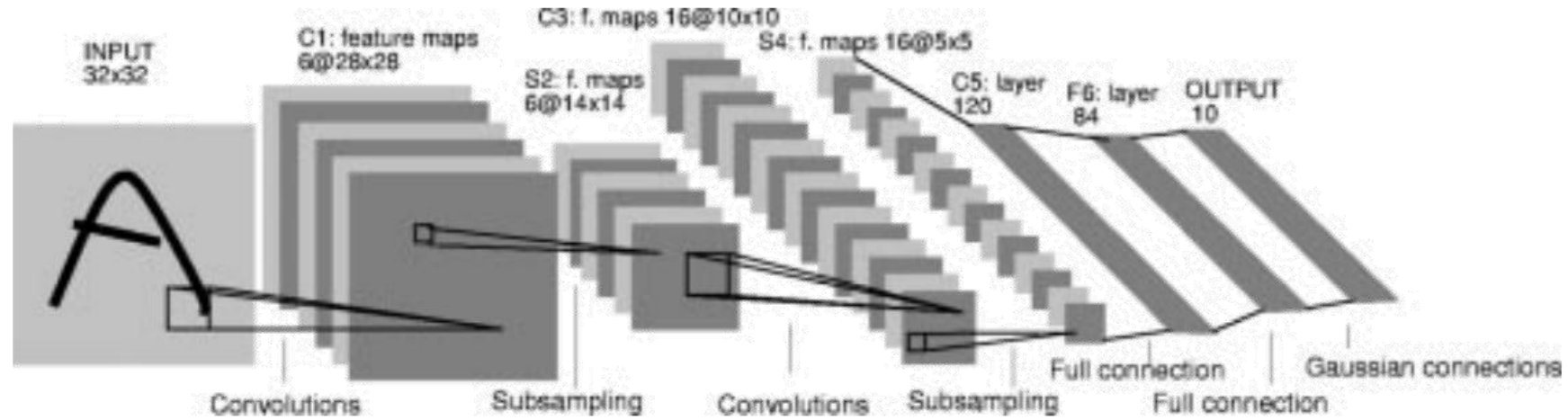


Computer Architecture - EE219
faculty.sist.shanghaitech.edu.cn

立志成才 报国裕民



Other PE Units



Memory

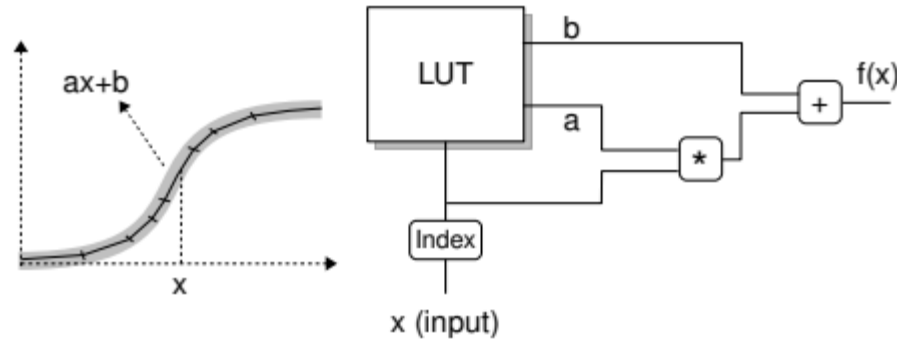
Matrix Multiply,
Conv. &
Accumulate

?

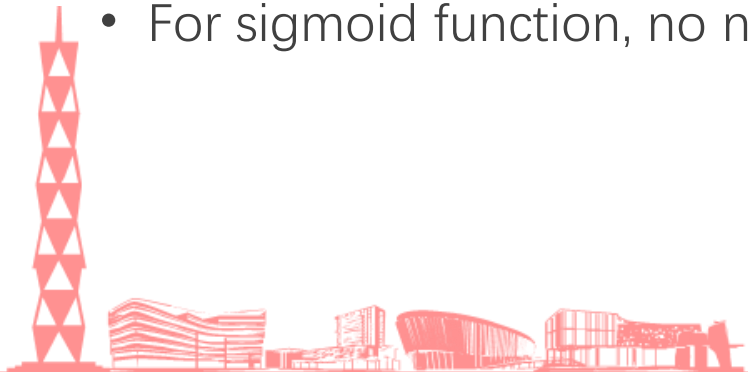


Non-linear Functions

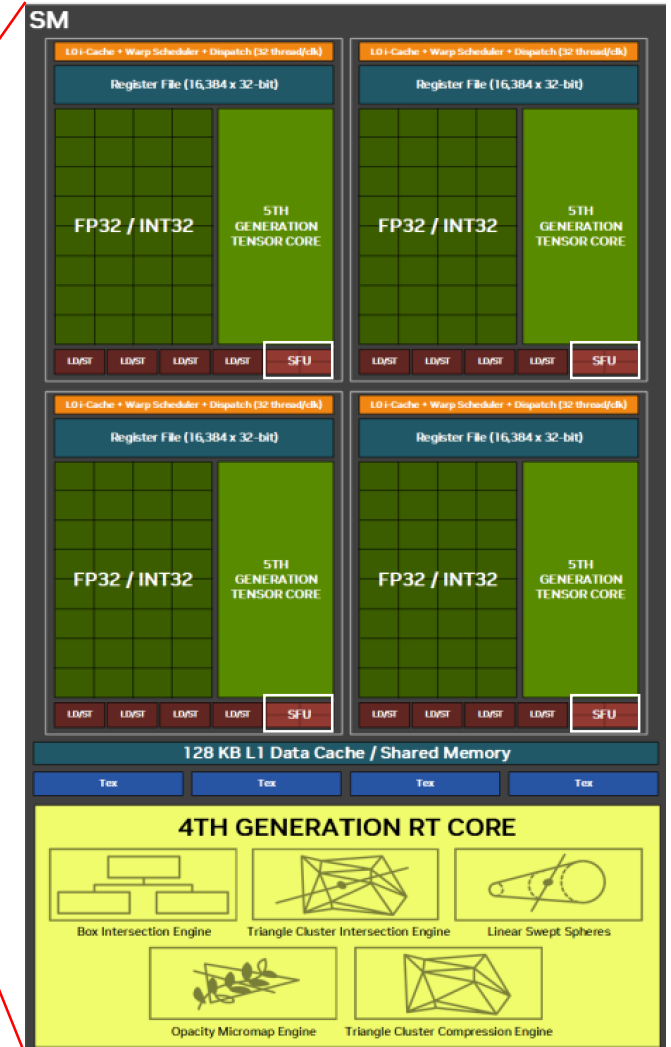
- Piecewise linear approximation in DianNao (sigmoid)



- For each piece, $f(x) = a_i \times x + b_i$ (fixed-point/integer operators)
- For sigmoid function, no noticeable impact on NN accuracy with 16 pieces.



Special Function Unit (SFU) in GPU

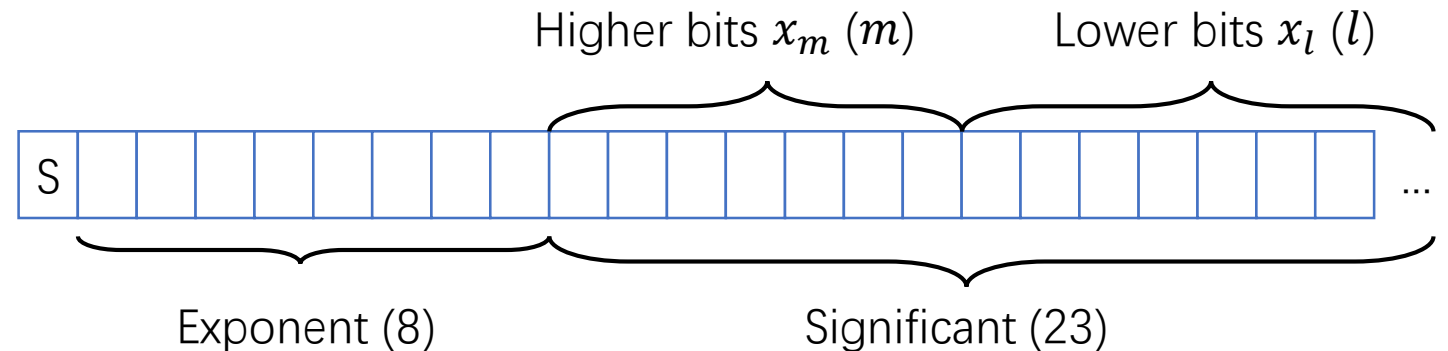


SM of Blackwell GPU

立志成才 报国裕民

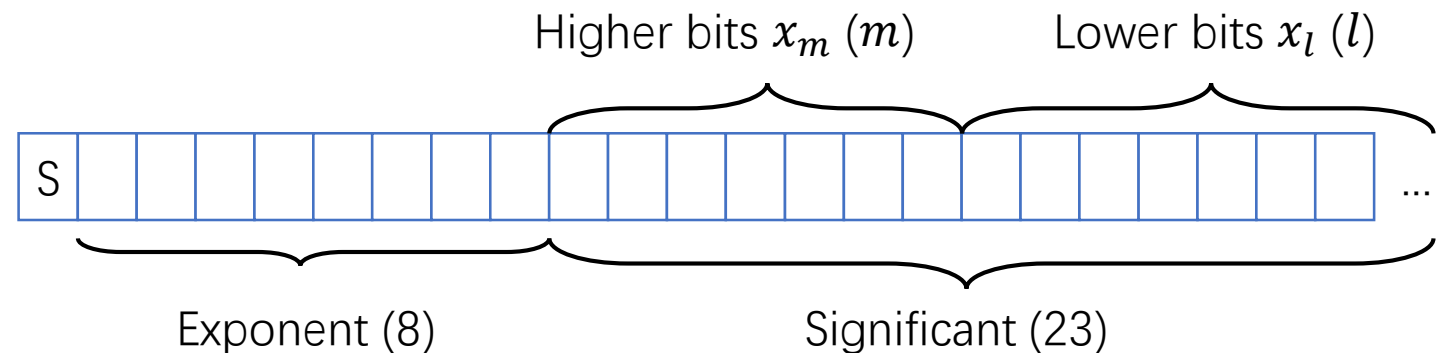
SFU in GPU

- Range reduction (pre-processing)
 - For sin/cos, 1/4 period is sufficient, $\sin X = \sin \left(\left(\frac{X}{2\pi} - \left\lfloor \frac{X}{2\pi} \right\rfloor \right) \times 2\pi \right)$
 - For log2X, $\log_2(XY) = \log_2(X) + \log_2(Y)$; $\log X = \log_2(X) / \log_2 e$
 - For exp, $\exp(X) = \text{power}_2(X \log_2 e)$, etc.
- Higher m bits as the index of an LUT
- Piecewise quadratic approximation (quadratic interpolation)
 - Transcendental function including sin/cos/log2/exp2/reciprocal/reciprocal square-root
 - Single-precision FP arithmetic in GPU, $f(x) \approx C_0 + C_1 x_l + C_2 x_l^2$

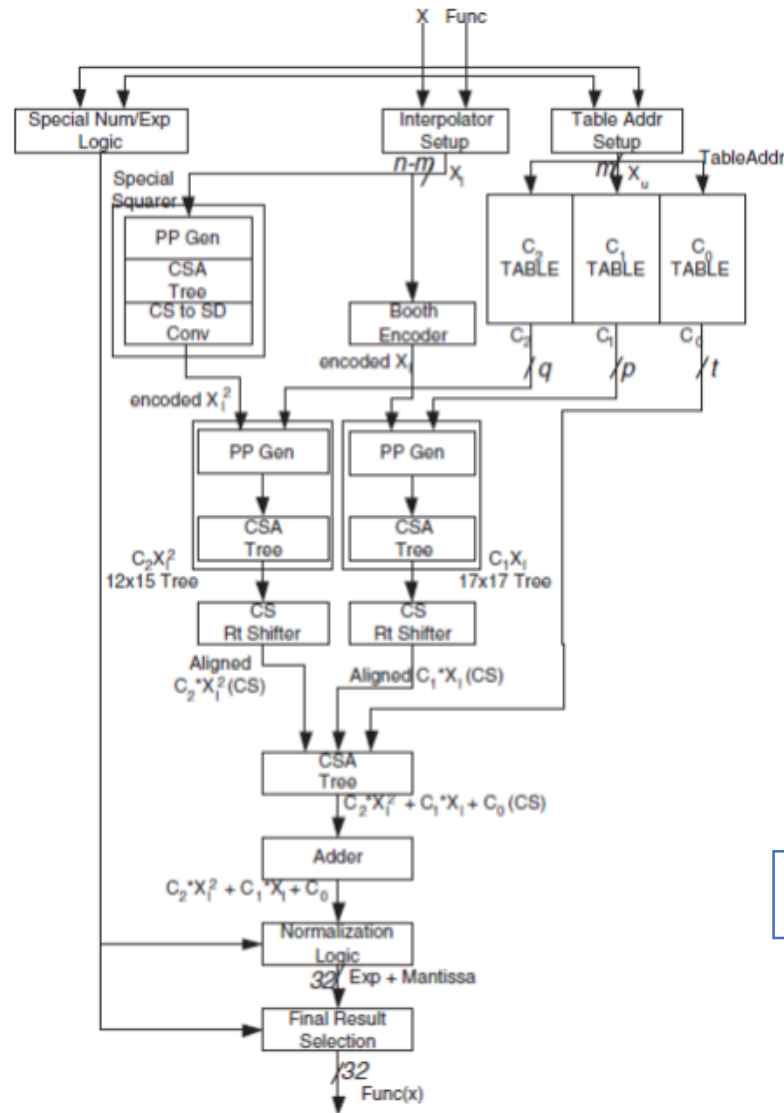


SFU in GPU (Cont'd)

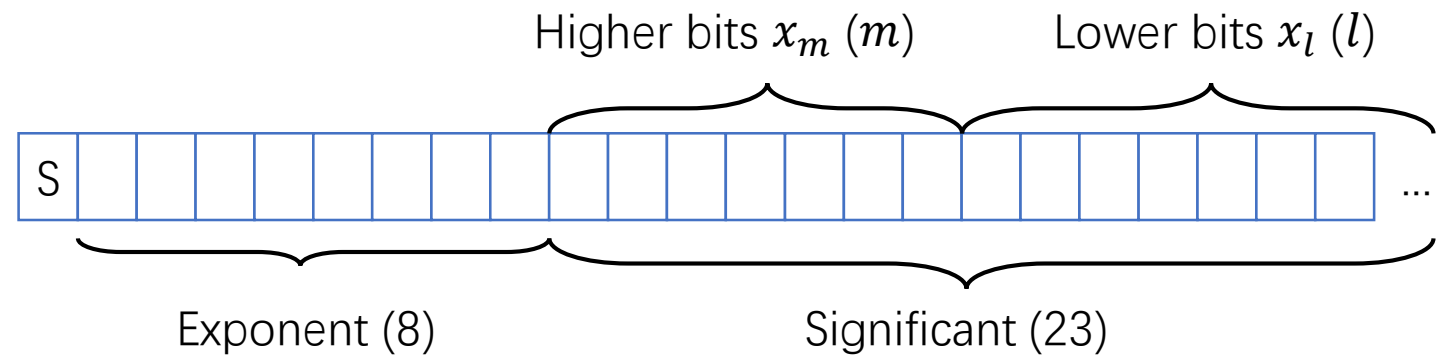
- Piecewise quadratic approximation (quadratic interpolation)
 - Transcendental function including sin/cos/log2/exp2/reciprocal/reciprocal square-root
 - Single-precision FP arithmetic in GPU, $f(x) \approx C_0 + C_1x_l + C_2x_l^2$
 - Coefficient generation
 - Use Remez exchange algorithm to compute **minimax** polynomial $a_0 + a_1x_l + a_2x_l^2$
 - Optimize the coefficient and quantize to t, p, q bits respectively, obtain C_0, C_1, C_2
 - Exhaustive search for optimal m, t, p, q
- Round/post-normalize



SFU Implementation



- Dedicated circuit to compute square
- Carry-save adders & booth encoders (for fast additions and multiplications)
- LUT implemented by ROM



SFU Implementation

Function	Input Interval	m	Configuration	Accuracy (good bits)	ulp error	% exactly rounded	monotonic	Lookup table size
$1/X$	$[1,2)$	7	26,16,10	24.02	0.98	87%	yes	6.50Kb
$1/\sqrt{X}$	$[1,4)$	6	26,16,10	23.40	1.52	78%	yes	6.50Kb
2^X	$[0,1)$	6	26,16,10	22.51	1.41	74%	yes	3.25Kb
$\log_2 X$	$[1,2)$	6	26,16,10	22.57	n/a	n/a	yes	3.25Kb
sin/cos	$[0, \pi/2]$	6	26,15,11	22.47	n/a	n/a	no	3.25Kb
Total								22.75Kb

Widths of C_0, C_1, C_2

CORDIC (COordinate Rotation Digital Computer)

- CORDIC (Coordinate Rotation Digital Computer)

- Sine/cosine

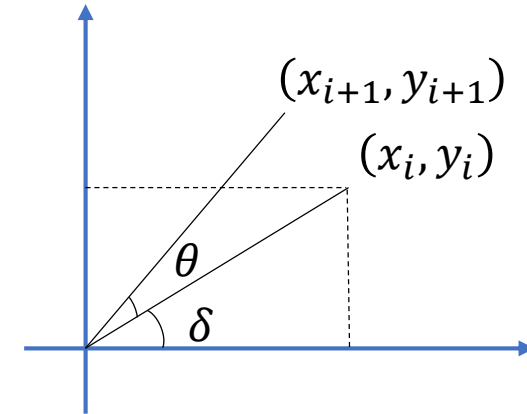
$$\sin(\delta + \theta) = \sin \delta \cos \theta + \cos \delta \sin \theta$$

$$\cos(\delta + \theta) = \cos \delta \cos \theta - \sin \delta \sin \theta$$

$$x_{i+1} = \cos(\delta + \theta)$$

$$y_{i+1} = \sin(\delta + \theta)$$

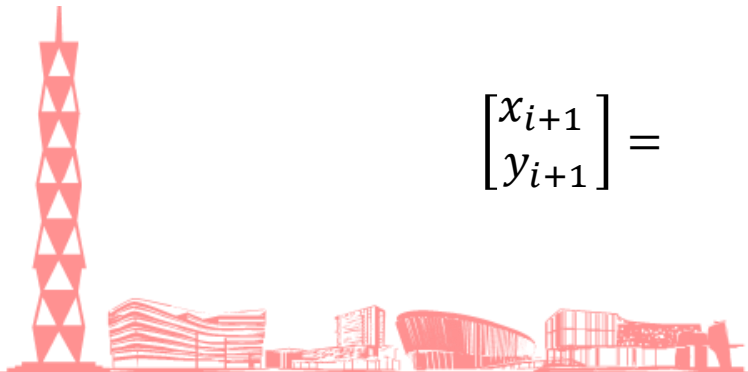
$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$



- Decompose an angle z

- $z_i = \sum_i \theta_i$

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$



CORDIC (Cont'd)

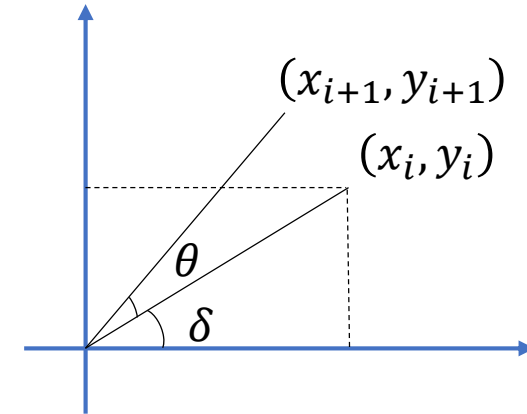
- Reduce multiplications: let $\tan \theta_i = \pm 2^{-i}$

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cos \theta_i \begin{bmatrix} 1 & -\tan \theta_i \\ \tan \theta_i & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$z_{i+1} = z_i \pm \theta_i$$

$$\theta_i = \text{atan}(2^{-i})$$

- Example: compute $\sin(53^\circ) \cos(53^\circ)$
- Iteration = 5



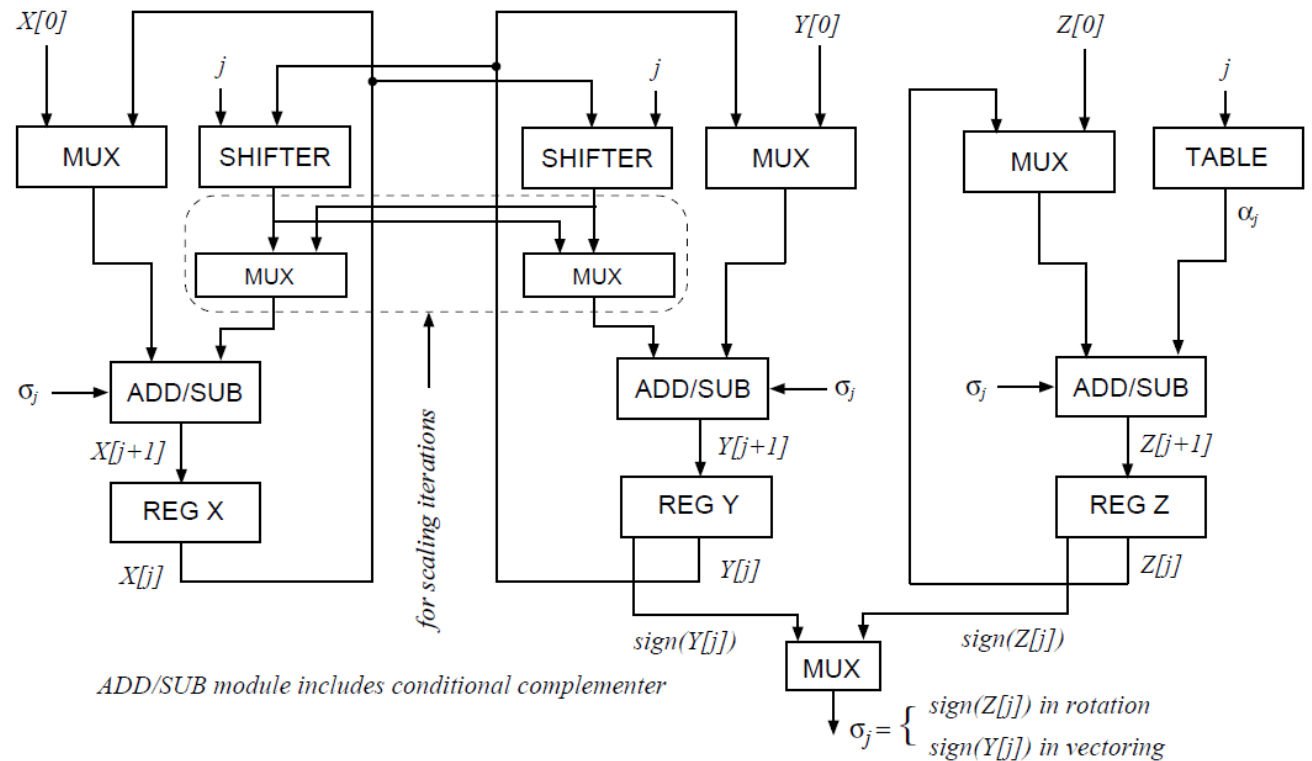
CORDIC Implementation

- Reduce multiplications: let $\tan \theta_i = \pm 2^{-i}$

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \cos \theta_i \begin{bmatrix} 1 & -\tan \theta_i \\ \tan \theta_i & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$z_{i+1} = z_i \pm \theta_i$$

$$\theta_i = \text{atan}(2^{-i})$$



Summary

- Various PE units (matrix multiplications/function approximations)
- Multiply-and-add/adder tree for matrix multiplications
- Dataflow & on-chip memory access
- Systolic arrays for matrix multiplications
- Function approximation using
 - Piecewise linear/quadratic function
 - CORDIC



Computer Architecture Review (Control Flow Architecture)

Instructor: Siting Liu

School of Information Science and Technology (SIST), ShanghaiTech University

Shanghai Institute of Semiconductor Innovation (SISI)

<https://faculty.sist.shanghaitech.edu.cn/liust/EE219/EE219.html>



Computer Architecture - EE219
faculty.sist.shanghaitech.edu.cn

立志成才 报国裕民

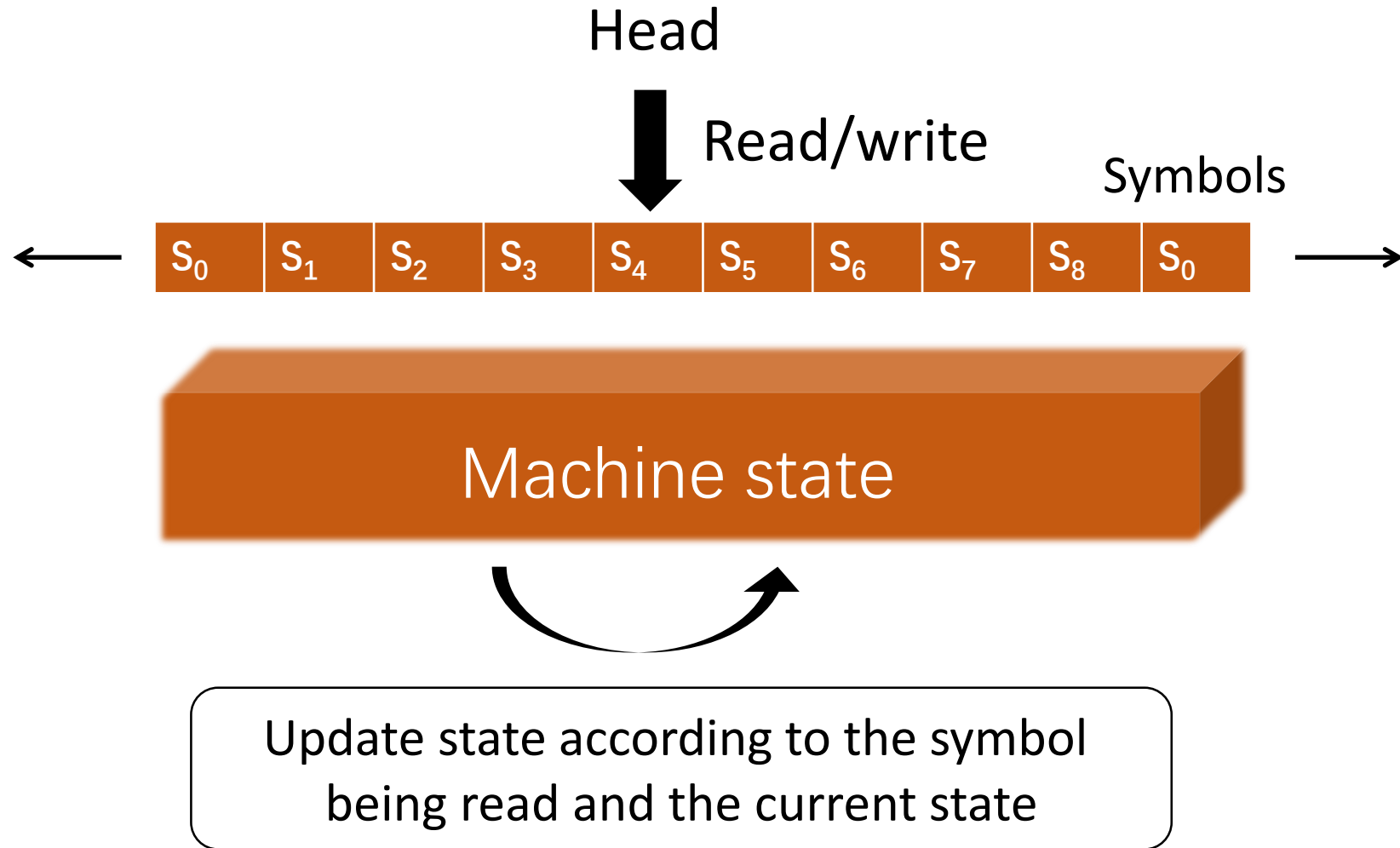


Objective

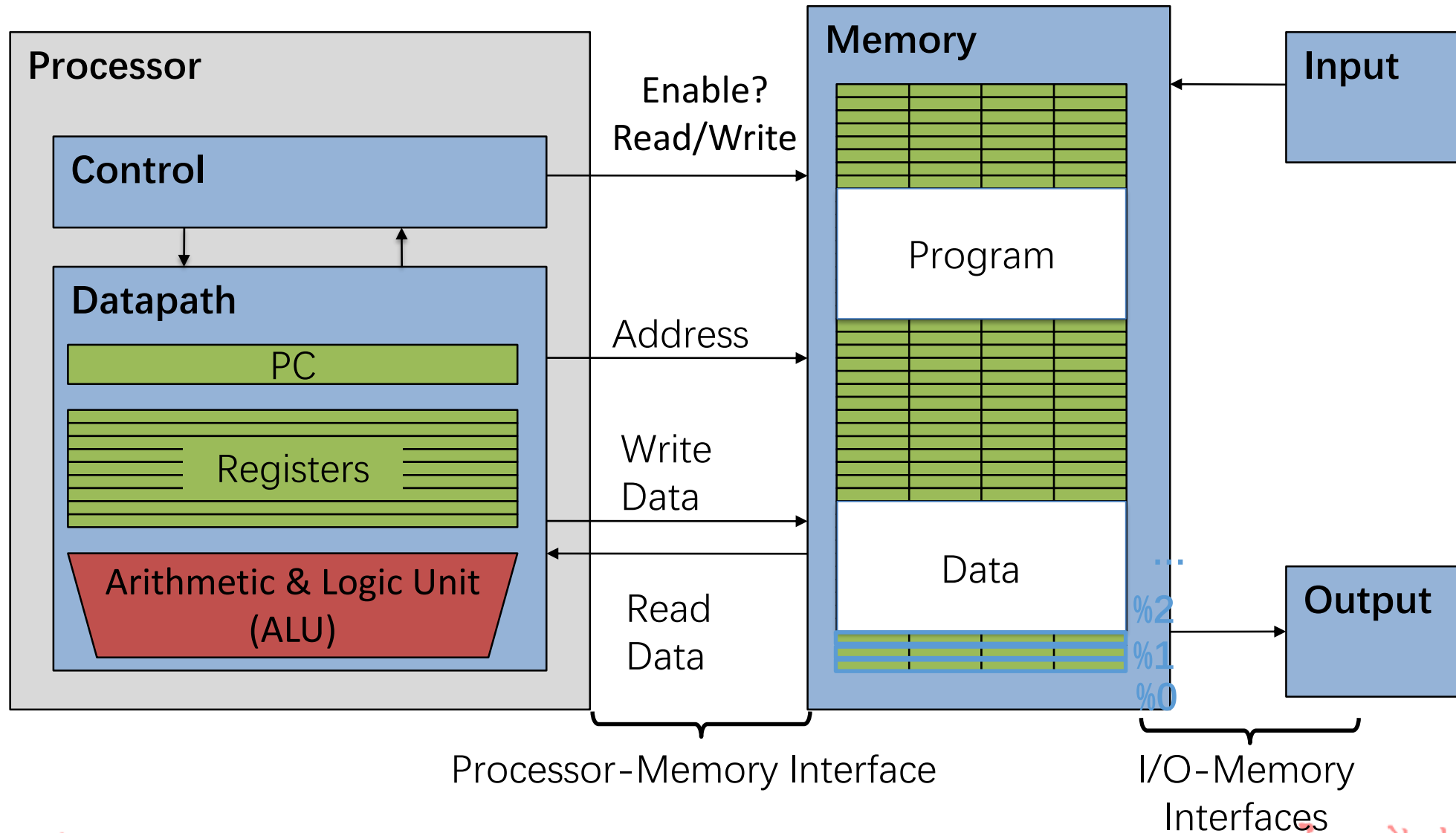
- Understand the design philosophy of conventional processors, e.g., CPU;
- Connect the algorithm to physical constraints of silicon;
- Integrate the accelerators we just designed into a computing system;
 - Make it a complete computer system, narrowed for NN workload;
 - Need software stack to make the design user-friendly;
 - Requires ISA as a hardware-software bridge;
- Study the techniques that can improve AI-chip design;



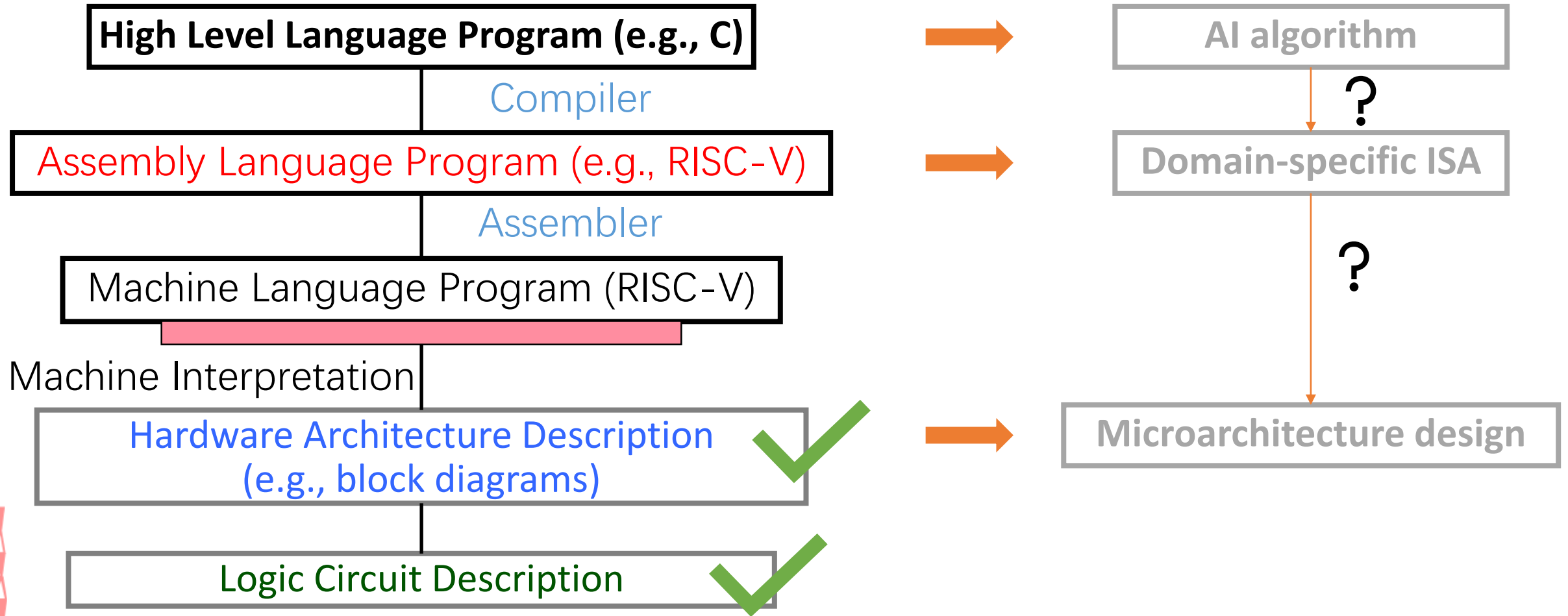
Control Flow Architecture – Turing Machine



Von Neumann Architecture



How does CPU work to Run a Program?



A Simple CPU Design

- A digital circuit that understands and executes instructions
- Requires ISA as the design manual, it decides
 - How memory is managed, such as addressing, load/store;
 - How to encode/decode an instruction or instruction format;
 - Registers, program counter and the other flags;



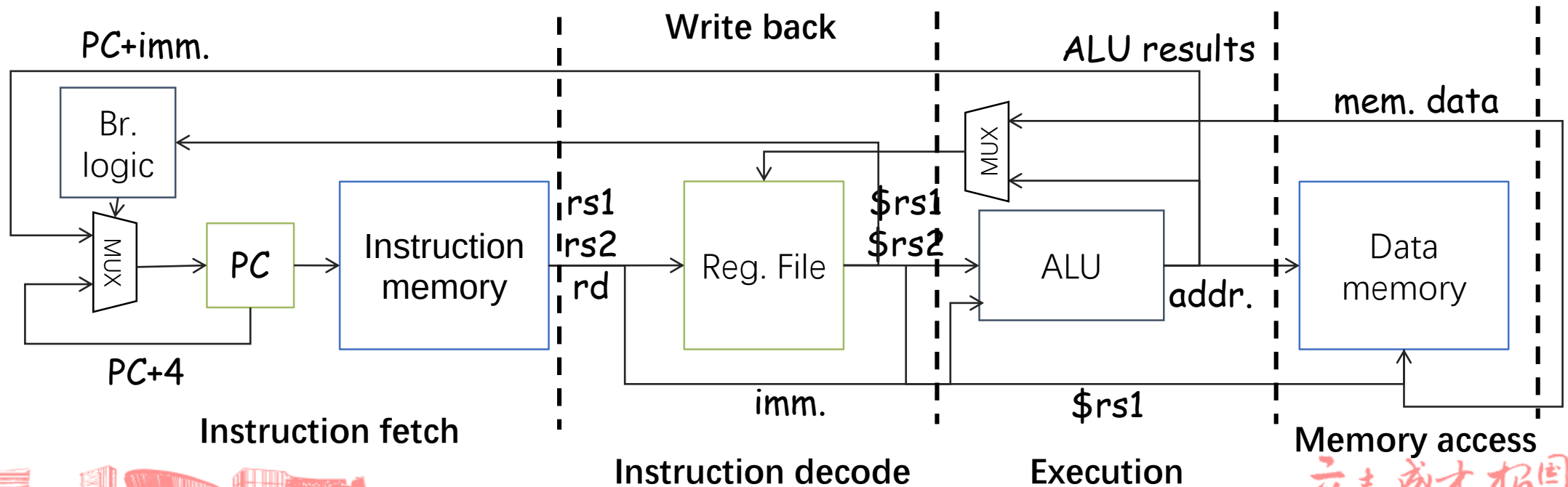
RISC-V for Example

- Memory is byte-addressable; load-store architecture;
- Fixed-instruction length, with each field represents certain information;
 - R-type for variable arithmetic and logic operations;
 - I-type for variable-constant arithmetic and logic operations and **Load** operations;
 - S-type for **Store** and **branch** operations;
 - (U)J-type for jump operations;
- 32 general-purpose registers (reg. file, logical), indexed by x0-x31 with x0=0
- Use program counter to indicate the address of the instruction being executed.
-



A Simple CPU that Executes RISC-V Instructions

- To implement 4 instructions
 - `add rd, rs1, rs2`; `lw rd, imm(rs1)`; `sw rs0, imm(rs1)`; `beq rs0, rs1, label`
- Basic components for (1) instruction fetch; (2) instruction decode and operand fetch; (3) execution; (4) memory access (if present); (5) write back (if present); (6) should also update PC for next instruction fetch.



Evaluation of Performance – Iron Law

- Performance

$$\text{Time} = \frac{\# \text{ Instruction}}{\text{Program}} \times \frac{\text{Instruction}}{\text{Cycle}} \times \text{Period.}$$

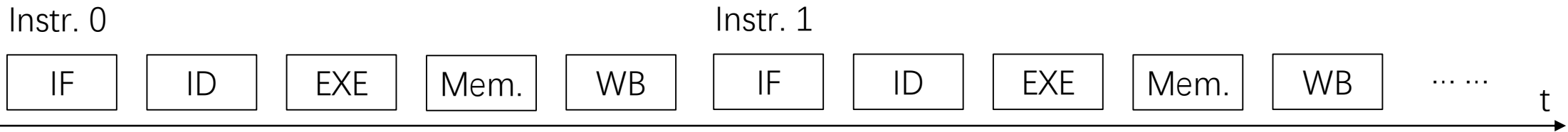
- In computer architecture, instruction-level parallelism, data-level parallelism and thread-level parallelism are employed to improve performance;



Instruction-level Parallelism (ILP)

- ILP can be achieved with pipelining and multi-issue

Non-pipelining



Pipelining

