CISC vs RISC

↳ CPU, have hundred of instructions for every possible situation

CISC

↳ Complex Instruction Set Computer

↳ CISC reduce number of instruction by having multiple operation in a single instruction

↳ goal is to complete a task in as few lines as possible

↳ MULT 2:3, 5:2

↳ 1. loads 2 value into separate register
2. Multiply the operand
3. Store in the appropriate register

↳ a = a * b

↳ Advantage :-

- do very little work to translate high-level language
- very little RAM is required
- build compux ins. directly into hardware

RISC

↳ Reduced Instruction Set Computer

↳ argue that not all instruction are used

↳ reduce instruction from 100 → ~40

↳ goal is to use simple instruction that can be executed in 1 clock cycle

↳ MULT 2:3, 5:2 { LOAD A, 2:3
LOAD B, 5:2
PROD A, B
STORE 2:3, A

same

↳ 1. loads 2 value into separate register
2. Multiply the operand
3. Store in the appropriate register

↳ a = a * b

↳ Advantage :-

- only one clock cycles to execute
- execute in same amount of time as CISC
- require less transcirtors → more room
- pipelining is possible
- reduce amount of work comp. must perform

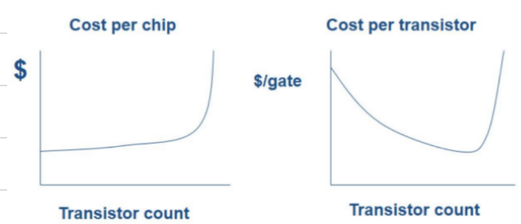
Which one is better?

↳ Nowdays hard to distinguish / boundary blured

↳ NO one cares

↳ In the 80s, either :-

- CISC, no on-chip cache
- RISC, w/ on chip cache



↳ w/ on-chip cache performs better

↳ RISC have no real advantage, just on-chip cache

↳ now CISC, w/ on-chip cache is possible

CISC	RISC
• Emphasis on hardware	• Emphasis on software
• Includes multi-clock complex instructions	• Single-clock, reduced instruction only
• Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	• Register to register: "LOAD" and "STORE" are independent instructions
• Small code sizes, high cycles per second	• Low cycles per second, large code sizes
• Transistors used for storing complex instructions	• Spends more transistors on memory registers