Microprocessors & Microcontrollers

: Arm Cortex M0+

(Using RP2040)

ES6

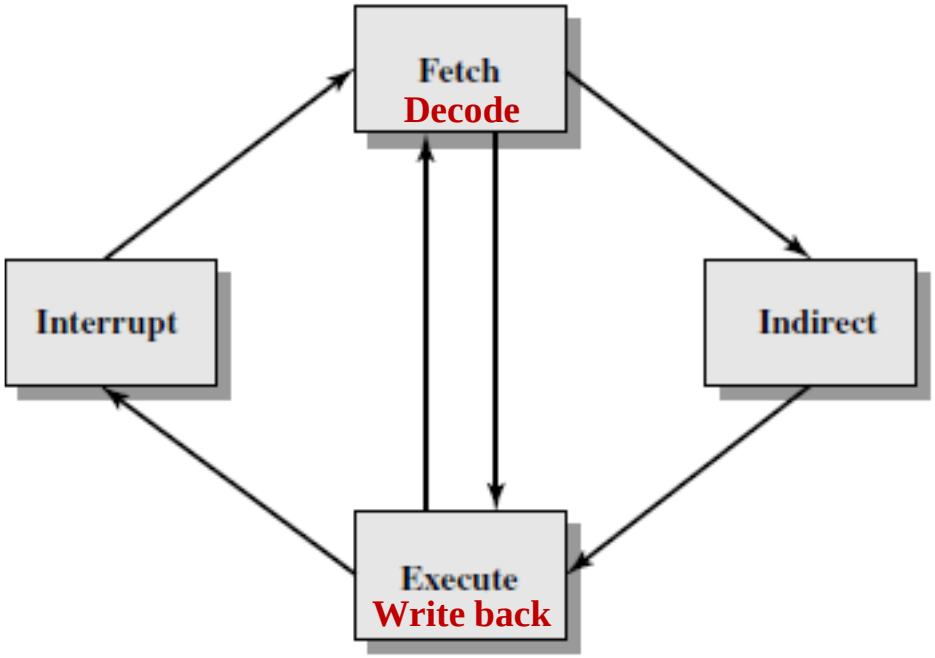
Instruction Cycle and Instruction Execution

Lecture 1.2.2: Focus

- Instruction Cycle
  - State Transition Diagram
- Instruction Execution (unpipelined)

Instruction Cycle

Instruction Cycle



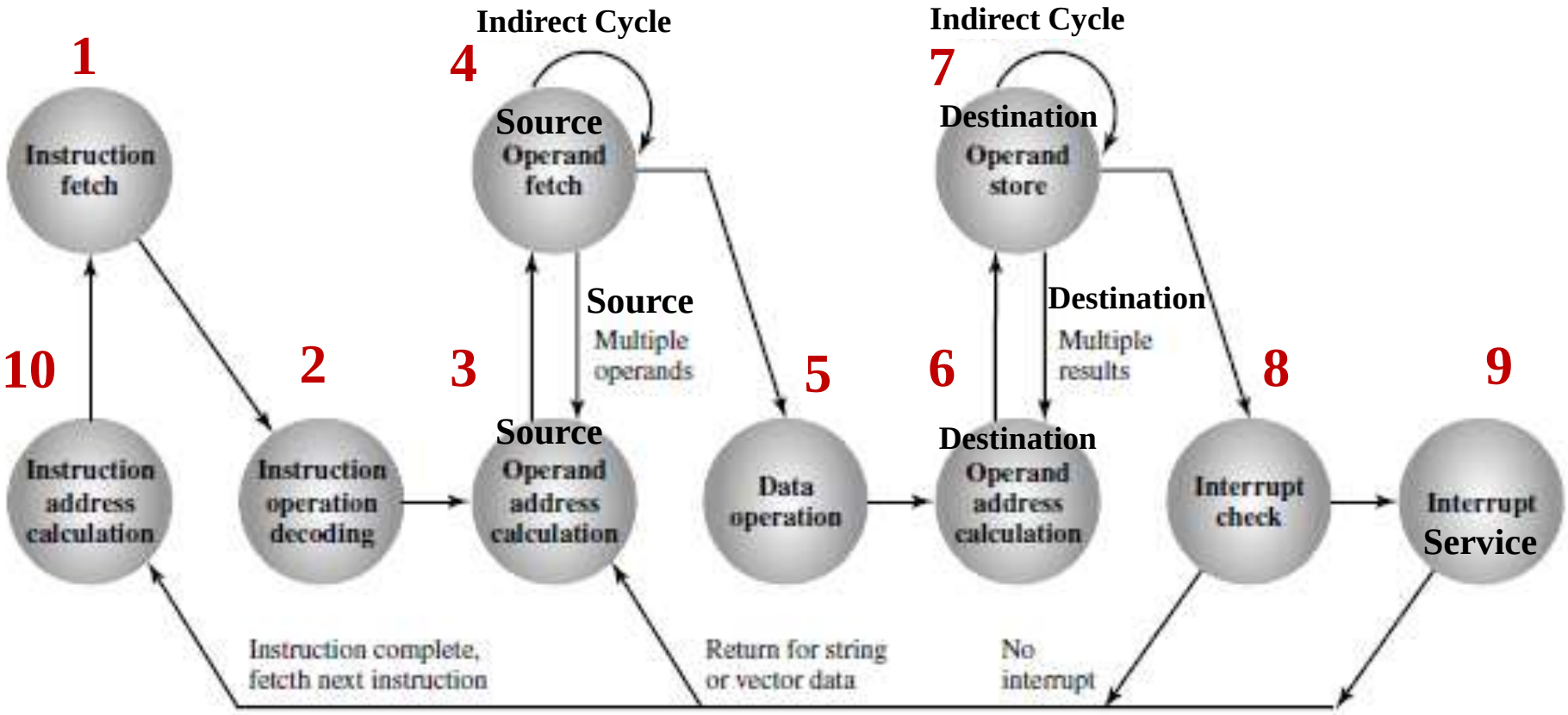
- **Fetch:** Read the next instruction from the memory into the processor.
- **Decode:** By decoding the opcode, the CPU understands the operation to be performed.
- **Indirect:** If the operands are from memory, initiate memory Read cycle (**indirect**), if not, read the operands from the registers (**direct**)
- **Execute:** Perform the indicated operation on the operands.
- **Write back:** Write the results back into a register.

- **Interrupt: After** the **completion** of **execution** of **every instruction**, if **interrupts** are **enabled** and an **interrupt** has **occurred**, **save** the **current CPU state** and **service** the **interrupt.**

ADD R1, R2, R3 **Direct addressing**
R1 ← R2 + R3

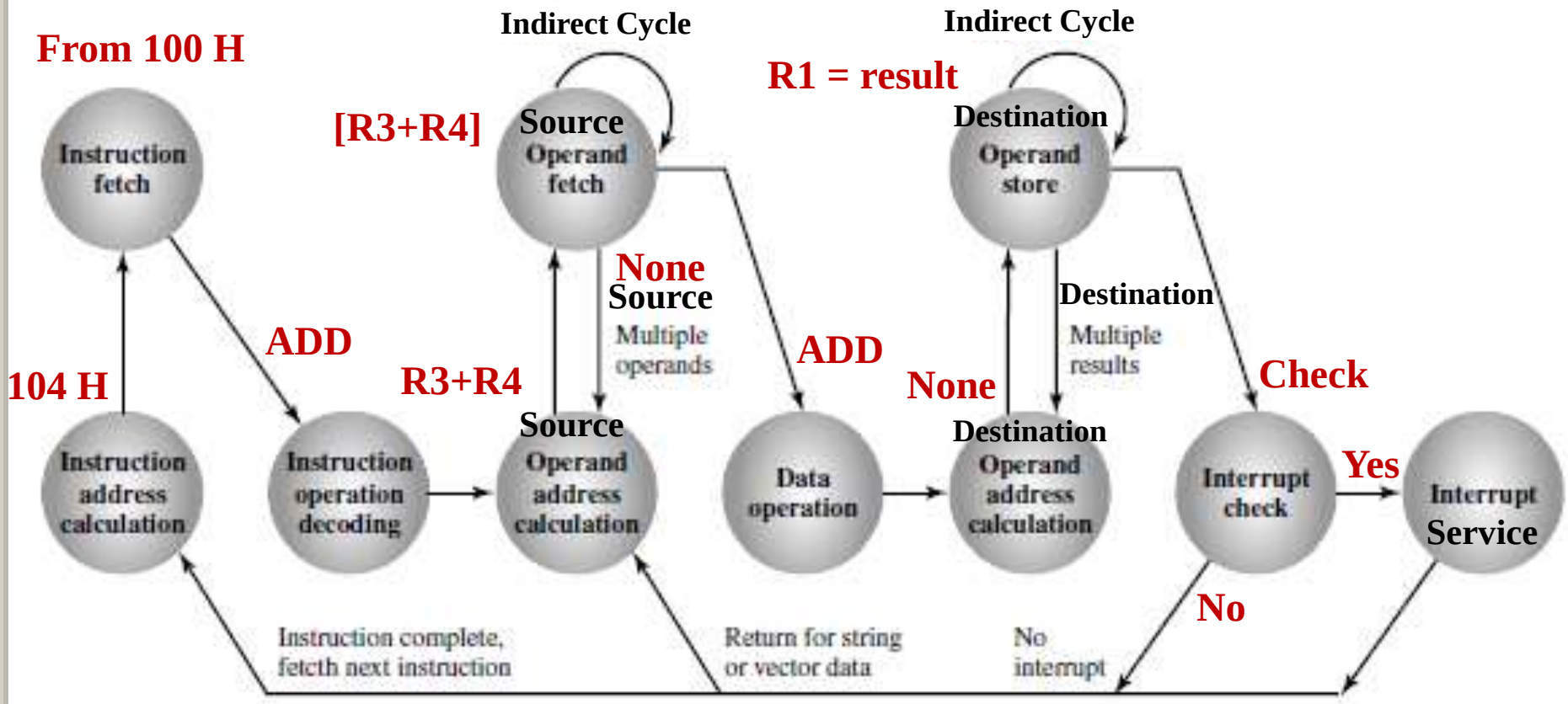
ADD R1, R2, [R3] **Indirect addressing**
R1 ← R2 + [R3]

Instruction Cycle: State Transition Diagram



Let us understand this better with an instruction next …

Instruction Cycle: ADD Instruction

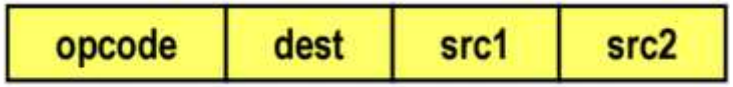


PC → 100H ADD R1, R2, [R3+R4]

New PC → 104H ……

Note: Normally, ALU is not used to compute the address of the next instruction, a dedicated adder takes care of it.

Instruction format: 32 bits wide (4 bytes)



Operation to be performed

R1 ← R2 + [R3+R4]

Instruction Execution: Unpipelined

(Many operations may take different CPU cycles based on the type of instructions)

- Instruction **Fetch**: Instruction is read from the program memory.
- **Decode**: Instruction is decoded to know the operation to be performed by the CPU.
- Depending upon the type of instruction the following operations are done.
  - **Operand fetch** (no operand or single operand or multiple operands).
  - Types of the operand fetch could either be direct or indirect .
    - **Direct**: One of the registers itself is an operand.
    - **Indirect**: Operand is in memory. Compute the address based on the addressing mode.
    - Then, an Indirect cycle is started to get the operand from the memory.
    - Operand is brought into the CPU's internal general purpose register.
- **Execute**: Operation is performed by the ALU.
- **Write back**: Results are stored back into a register or memory if the instruction is a store.
- All these operations are done over a few cycles before the next instruction is brought into the CPU. – Unpipelined execution.

Summary

- Instruction Cycle
  - State Transition Diagram
- Instruction Execution (unpipelined)