Computer Organization & Architecture

Chapter 2 – RISC and CISC Styles

Zhang Yang 张杨
cszyang@scut.edu.cn

Autumn 2021

Content of this lecture

- 2.10 CISC Instruction Sets
  - Additional Addressing Modes
  - Condition Codes
- 2.11 RISC and CISC Styles

CISC Instruction Sets (1)

- CISC instruction sets are not constrained to the *load/store architecture*, in which arithmetic and logic operations can be performed only on operands that are in processor registers.
- CISC instructions do not necessarily have to fit into a single word. Some instructions may occupy a single word, but others may span multiple words.
- Most arithmetic and logic instructions use the *two-address* format.
  - Operation *destination*, *source*
  - Example: Add B, A
    - Performs the operation B $\leftarrow$ [A] + [B] on memory operands.

CISC Instruction Sets (2)

- The Move instruction includes the functionality of the Load and Store instructions.
  - Move destination, source
  - Example C = A + B
    - Move C, B
    - Add C, A
  - In some CISC processors one operand may be in the memory but the other must be in a register.
    - Move R*i*, A
    - Add R*i*, B
    - Move C, R*i*

CISC Instruction Sets (3)

- Autoincrement and Autodecrement Mode
  - Autoincrement Mode
    - The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list.
    - (Ri)+
    - EA = [Ri] Increment Ri
    - Useful for adjusting pointers in loop body:

```
Add            SUM, (Ri)+
MoveByte       (Rj)+, Rk
```

  - Increment by 4 for words, and by 1 for bytes

CISC Instruction Sets (4)

- Autoincrement and Autodecrement Mode (ctd.)
  - Autodecrement Mode
    - The contents of a register specified in the instruction are first automatically decremented and are then used as the effective address of the operand.
    - $-(Ri)$
    - Use autoinc. & autodec. for stack operations:

```
Move    –(SP), NEWITEM        (push)
Move    ITEM, (SP)+           (pop)
```

CISC Instruction Sets (5)

- Relative Mode
  - The effective address is determined by the Index mode using the program counter in place of the general-purpose register Ri.
  - EA = [PC] + X, X is a signed number.
  - Usage
    - Access data operand.
    - Specify the target address in branch instructions.
      - Example   Branch > 0   Loop
        - The branch target location can be computed by specifying it as an offset from the current value of the program counter.

CISC Instruction Sets (6)

- Condition Codes
  - Processor can maintain information on results to affect subsequent conditional branches.
  - Results from arithmetic/comparison & Move.
  - Condition code flags in a status register:

N	(negative)	1 if result negative, else 0
Z	(zero)	1 if result zero, else 0
V	(overflow)	1 if overflow occurs, else 0
C	(carry)	1 if carry-out occurs, else 0

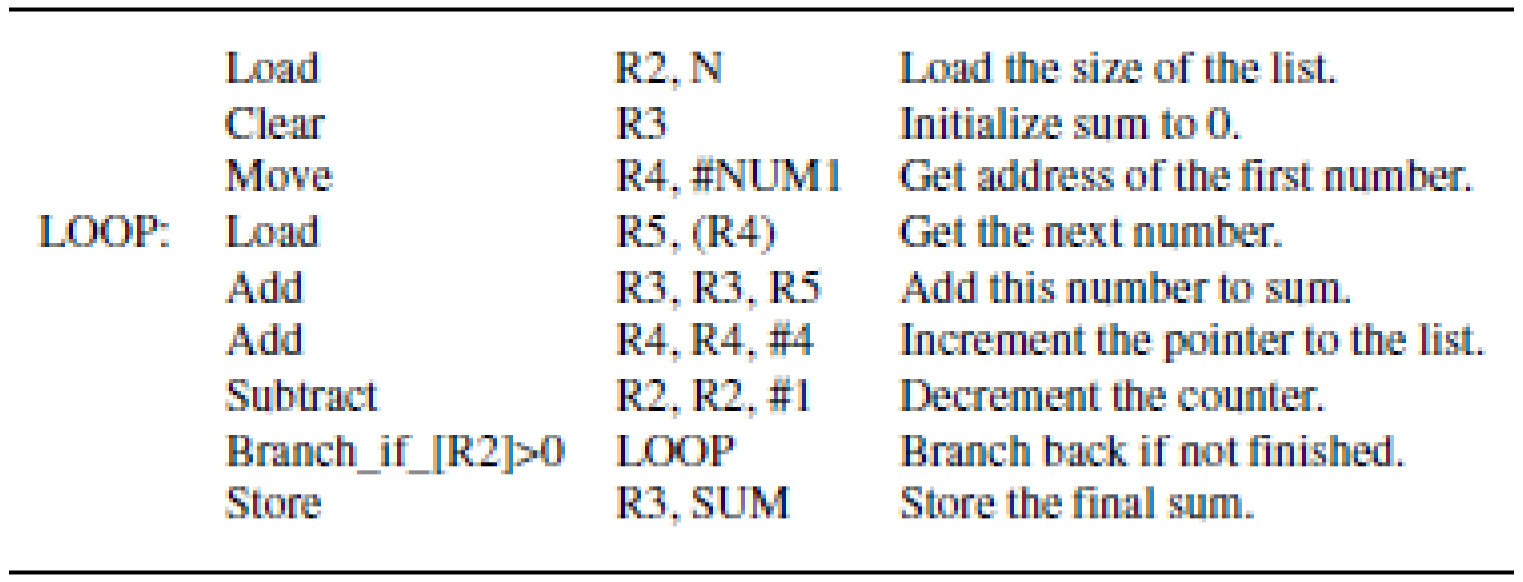
CISC Instruction Sets (7)

- Branches Using Condition Codes
  - CISC branches check condition code flags.
  - For example, decrementing a register causes N and Z flags to be cleared if result is *not* zero.
  - A branch to check logic condition $N + Z = 0$:

    Branch>0   LOOP
  - Other branches test conditions for $<, =, \neq, \leq, \geq$
  - Also Branch_if_overflow and Branch_if_carry.
  - Consider CISC-style list-summing program (next page)

CISC Instruction Sets (8)

- Branches Using Condition Codes (ctd.)

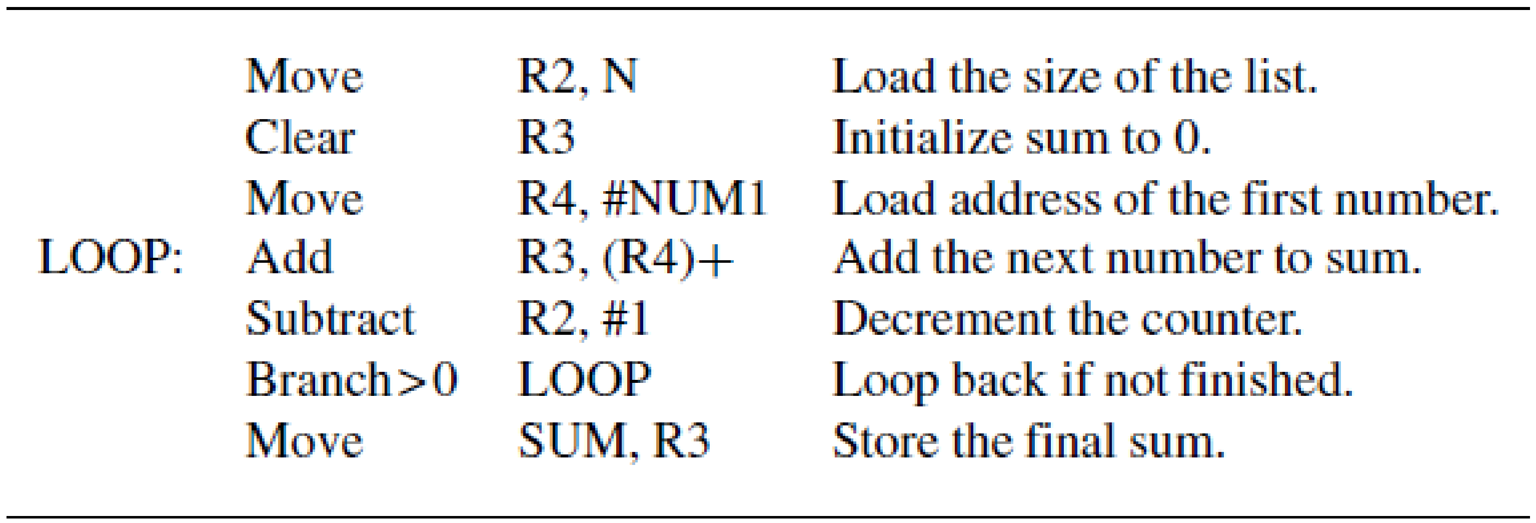


```
         Load               R2, N         Load the size of the list.
         Clear              R3            Initialize sum to 0.
         Move               R4, #NUM1     Get address of the first number.
LOOP:    Load               R5, (R4)      Get the next number.
         Add                R3, R3, R5    Add this number to sum.
         Add                R4, R4, #4    Increment the pointer to the list.
         Subtract           R2, R2, #1    Decrement the counter.
         Branch_if_[R2]>0   LOOP          Branch back if not finished.
         Store              R3, SUM       Store the final sum.
```

Figure 2.8 Use of indirect addressing in the program of Figure 2.6.

CISC Instruction Sets (9)

- Branches Using Condition Codes (ctd.)



```
         Move        R2, N        Load the size of the list.
         Clear       R3           Initialize sum to 0.
         Move        R4, #NUM1    Load address of the first number.
LOOP:    Add         R3, (R4)+    Add the next number to sum.
         Subtract    R2, #1       Decrement the counter.
         Branch>0    LOOP         Loop back if not finished.
         Move        SUM, R3      Store the final sum.
```

Figure 2.26 A CISC version of the program of Figure 2.8.

RISC and CISC Styles (1)

- RISC characteristics include:
    - Simple addressing modes
    - All instructions fitting in a single word
    - Fewer total instructions
    - Arithmetic/logic operations on registers
    - Load/store architecture for data transfers
    - More instructions executed per program
- Simpler instructions make it easier to design faster hardware (e.g., use of pipelining)

RISC and CISC Styles (2)

- CISC characteristics include:
  - More complex addressing modes
  - Instructions spanning more than one word
  - More instructions for complex tasks
  - Arithmetic/logic operations on memory
  - Memory-to-memory data transfers
  - Fewer instructions executed per program
- Complexity makes it somewhat more difficult to design fast hardware, but still possible

RISC and CISC Styles (3)

- Before the 1970s, all computers were of CISC type.
  - Move the complexity from the software level to the hardware level.
- RISC-style designs emerged as an attempt to achieve very high performance by making the hardware very simple.
- CISC and RISC implementations are becoming more and more alike. Many of today’s RISC chips support as many instructions as yesterday's CISC chips. And today's CISC chips use many techniques formerly associated with RISC chips.

RISC and CISC Styles (4)

- CISC vs RISC

CISC	RISC
Complex instructions require multiple cycles	Reduced instructions take 1 cycle
Many instructions can reference memory	Only Load and Store instructions can reference memory
Instructions are executed one at a time	Uses pipelining to execute instructions
Few general registers	Many general registers

Summary

- CISC Addressing Modes
  - 了解Autoincrement Mode
  - 了解Autodecrement Mode
  - 理解Relative Mode，掌握这种寻址方式有效地址的计算。
  - 理解condition code: N,Z, C, V
- 掌握RISC Characteristics
- 掌握CISC Characteristics