RISC and CISC Instruction Sets

- One of the most important characteristics that distinguish different computers is the nature of their instructions
- Two approaches in the design of instruction sets for modern computers
  - Reduced Instruction Set Computers(RISC)
  - Complex Instruction Set Computer(CISC)

RISC

- The restriction that each instruction must fit into a single word that reduces the complexity and the number of different types of instructions that may be included in the instruction set of a computer.
- Such computers are called Reduced Instruction Set Computers (RISC)
- Example-CDC 6600 (1964), ARM

RISC Features

- Simple addressing modes
- All instructions fitting in a single word
- Fewer instructions in the instruction set, as a consequence of simple addressing modes
- Arithmetic and logic operations that can be performed only on operands in processor registers

RISC Features

- Load/store architecture that does not allow direct transfers from one memory location to another; such transfers must take place via a processor register
- Simple instructions that are conducive to fast execution by the processing unit using techniques such as pipelining
- Programs that tend to be larger in size, because of more, but simpler instructions are needed to perform complex tasks

RISC Instructions

- Memory operands are accessed only using Load and Store instructions
- Example: To execute C=A+B, we use three registers R2,R3,R4 ,the four instructions required are:

  Load R2, A

  Load R3, B

  Add R4, R2, R3

  Store R4, C

RISC Instruction

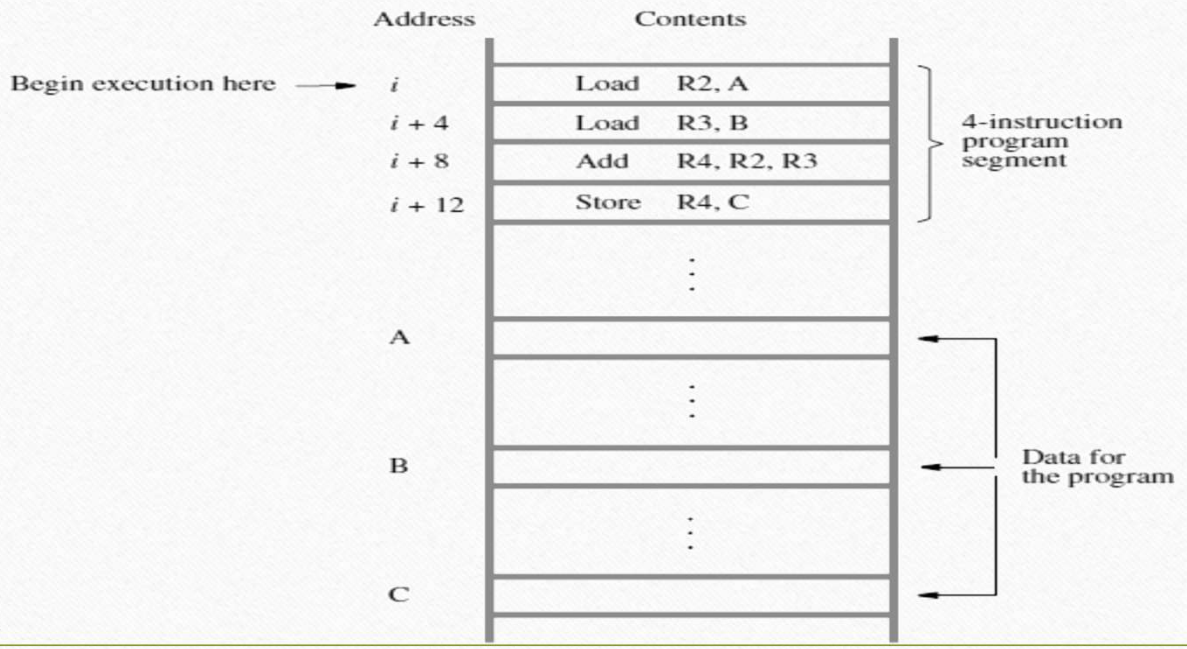

Figure: Instruction Execution and Straight-line Sequencing

RISC-Branching

- Consider the task of adding a list of *n* numbers
- Two Approaches
- 1) The addresses of the memory locations containing the *n* numbers are symbolically given as NUM1, NUM2, . . . , NUM*n*, and separate Load and Add instructions are used to add each number to the contents of register R2
- After all the numbers have been added, the result is placed in memory location SUM.

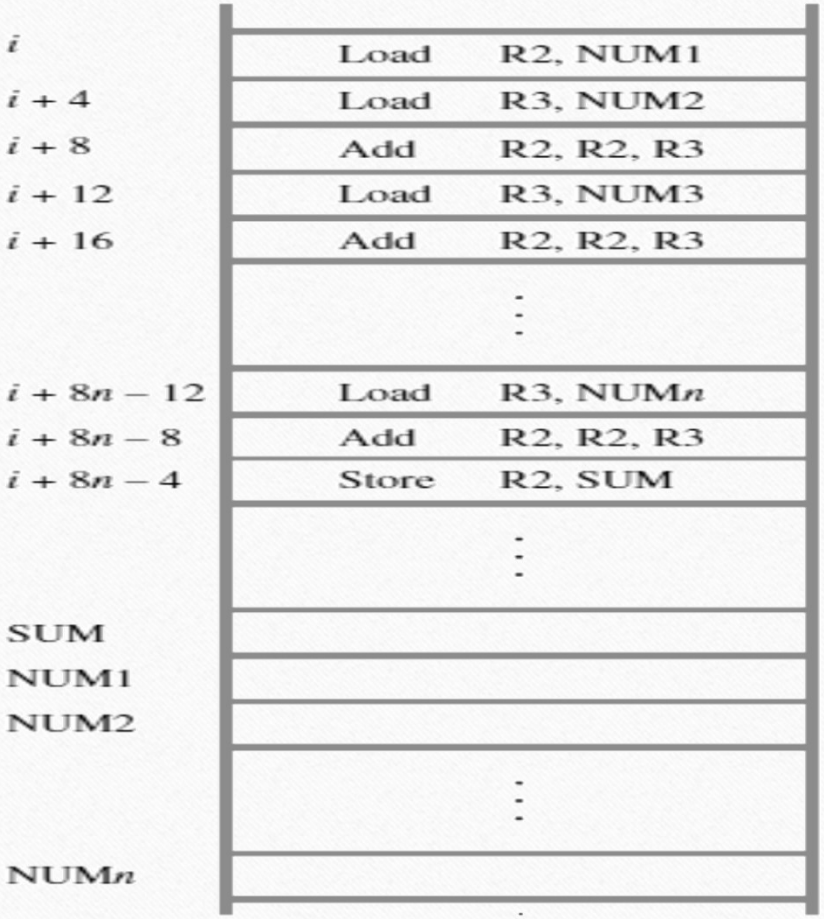

Figure: Adding list of numbers without branch

RISC-Branching

- 2) It is possible to implement a program loop in which the instructions read the next number in the list and add it to the current sum.
- To add all numbers, the loop has to be executed as many times as there are numbers in the list

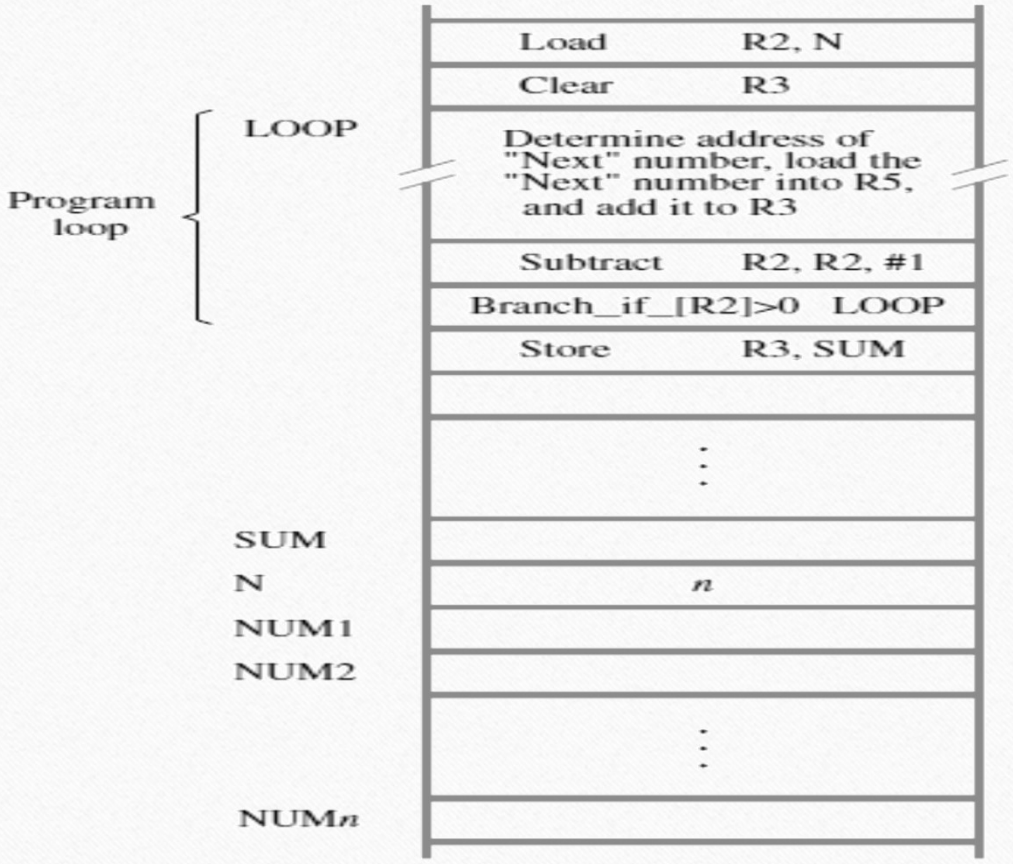

Figure: Adding list of numbers with branch

RISC –Addressing Mode

Name	Assembler syntax	Addressing function
Immediate	#Value	Operand = Value
Register	R*i*	EA = R*i*
Absolute	LOC	EA = LOC
Register indirect	(R*i*)	EA = [R*i*]
Index	X(R*i*)	EA = [R*i*] + X
Base with index	(R*i*,R*j*)	EA = [R*i*] + [R*j*]

EA = effective address
Value = a signed number
X = index value

```
        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: Use of Indirect address in addling list of N numbers

CISC(Complex Instruction Set Computer)

- Traditional approach
- Main features :
- A large number of instructions—typically from 100 to 250 instructions.
- Some instructions that perform specialized tasks and are used infrequently
- A large variety of addressing modes—typically from 5 to 20 different modes(R-R,R-M,M-M,Indexed,Indirect)
- Variable-length instruction formats

CISC(Complex Instruction Set Computer)

- Ease of mapping high-level language statement to machine code is possible in CISC, but instruction decoding and control unit design are much more complex
- In the variable length instruction, pipeline implementation will also be quite complex
- Example
  - IBM 360/370 (1960-70)
  - VAX-11/780 (1970-80)
  - Intel x86/Pentium (1985-Present)

CISC(Complex Instruction Set Computer)

- 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
- Instructions do not necessarily have to fit into a single word. Some instructions may occupy a single word, but others may span multiple words
- Instructions in modern CISC processors typically do not use a three-address format

CISC(Complex Instruction Set Computer)

- Most arithmetic and logic instructions use the two-address format:

  Operation destination, source

- An Add instruction of this type is Add B, A

  which performs the operation B ← [A] + [B] on memory operands

CISC(Complex Instruction Set Computer)

- Consider again the task of adding two numbers C = A + B where all three operands may be in memory locations
- Obviously, this cannot be done with a single two-address instruction. The task can be performed by using another two-address instruction that copies the contents of one memory location into another
- Such an instruction is Move C, B which performs the operation C←[B], leaving the contents of location B unchanged

CISC(Complex Instruction Set Computer)

- The operation C←[A] + [B] can now be performed by the two-instruction sequence

  Move C, B

  Add C, A

- Observe that by using this sequence of instructions the contents of neither A nor B locations are overwritten.

CISC(Complex Instruction Set Computer)

- In some CISC processors one operand may be in the memory but the other must be in a register.
- In this case, the instruction sequence for the required task would be

  Move Ri, A

  Add Ri, B

  Move C, Ri

CISC-Addressing Modes

- Most CISC processors have all of the five basic addressing modes —Immediate, Register, Absolute, Indirect, and Index
- Three additional addressing modes are often found in CISC processors
- ***Autoincrement and Autodecrement Modes*** - There are two modes that are particularly convenient for accessing data items in successive locations in the memory and for implementation of stacks.
- ***Relative mode***—The effective address is determined by the Index mode using the program counter in place of the general-purpose register R*i*.

CISC-Addressing Modes

- 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 operand in memory
- Autoincrement mode is written as *(Ri)+*

CISC-Addressing Modes

- We denote the Autoincrement mode by putting the specified register in parentheses, to show that the contents of the register are used as the effective address, followed by a plus sign to indicate that these contents are to be incremented after the operand is accessed
- Computers that have the Autoincrement mode automatically increment the contents of the register by a value that corresponds to the size of the accessed operand
- Thus, the increment is 1 for byte-sized operands, 2 for 16-bit operands, and 4 for 32-bit operands.

CISC-Addressing Modes

- 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.
- We denote the Autodecrement mode by putting the specified register in parentheses, preceded by a minus sign to indicate that the contents of the register are to be decremented before being used as the effective address
- Thus, we write $-(Ri)$
- In this mode, operands are accessed in descending address order

CISC-Addressing Modes

- Example: Main reason for autoincrement and autodecrement mode is to implement a stack structure.
- Instead of needing two instructions to push a new item on the stack

  Subtract SP, #4

  Move (SP), NEWITEM we can use just one instruction Move −(SP), NEWITEM
- Similarly, instead of needing two instructions to pop an item from the stack
  - Move ITEM, (SP)
  - Add SP, #4 , we can use just have Move ITEM, (SP)+

CISC-Addressing Modes

- Relative Mode:
- Some computers have a version of this mode in which the program counter, PC, is used instead of a general-purpose register
- Then, X(PC) can be used to address a memory location that is X bytes away from the location presently pointed to by the program counter
- Since the addressed location is identified *relative* to the program counter, which always identifies the current execution point in a program, the name Relative mode is associated with this type of addressing.

CISC-Example

```
        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.
```