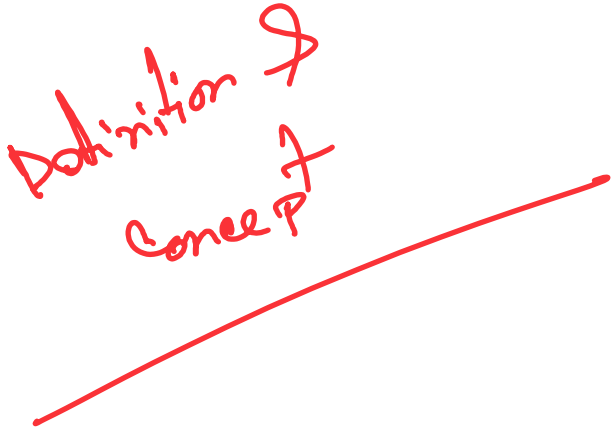


Lecture - 06

Classifying Instruction Set

- **CISC = Complex Instruction Set Computer**
  - Large number of instruction format in instruction set architecture (ISA)
  - Large number of addressing schemes and instruction types
  - Varying size of instructions
  - Multiple cycle instruction execution

- **RISC = Reduced Instruction Set Computer**
  - Small number of fast instruction format
  - Fixed instruction format, fixed length
  - Fast and single cycle instruction execution
  - Large number of general purpose registers
  - Typical for load/store architectures

RISC Vs CISC

- CISC (complex instruction set computer)
  - VAX, Intel X86, Motorolla 680X0, etc

- RISC (reduced instruction set computer)
  - MIPS, DEC Alpha, SUN Sparc, IBM 801,ARM6, etc

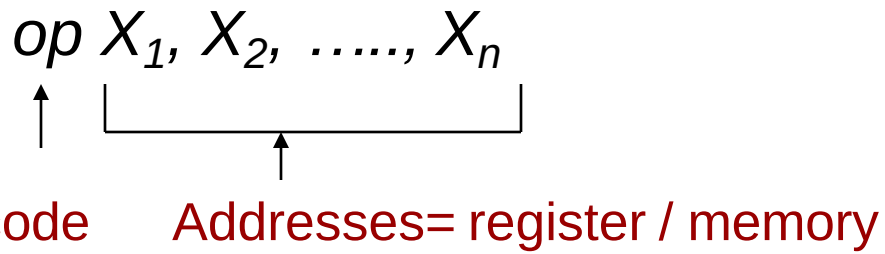
RISC Vs CISC

CISC	RISC
Variable length instruction	Single word instruction
Large number of format	Small number of format
Memory operands	Load/store architecture
Complex operations	Simple operations

Instruction Format

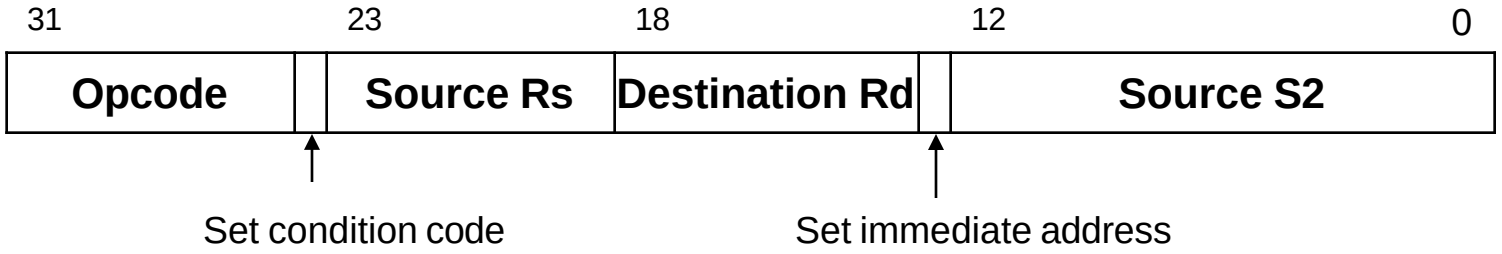
An instruction specifies

- the operation to be carried out by the processor.
- the set of operands to be used in the operation. It includes both the input data of the operation and the result that are produced.
- In assembly language notation, an instruction format can be written as



Instruction Format for RISC1

As memory addressing is restricted to load and store instructions, so the operands of most instructions are register addresses, Which are short and easy to accommodate in a one-word format.



The register-to-register operation form

Rd:=F(Rs, S2)

Rd – Destination register, Rs – First source register, S2 – Second source register

If bit 13 = 1, then S2 is interpreted as an immediate address

Operand Extension

- A CPU is designed to process data words and addresses of one specific length.
- Instruction contains operands fields that are shorter than standard word size.
- This problem is unavoidable in RISC instructions.
- Two methods to extend short operand values to full-size, signed or unsigned numbers:
  1. Sign Extension
  2. Zero Extension

Sign Extension

- Suppose a short m-bit, twos-complement number N is used in an n-bit arithmetic operation. This technique replicates the leftmost bit s of the short operand N, which corresponds to its sign bit, n-m times.

- Example: N = 10101 01010101

  $N_{Sign\text{-}extended}$ =

  1111111111111111110101010101

  n-m = 19 13

Sign Extension

- N = 10101 01010101= -2646_{10}
- It maintains a number’s correct sign and magnitude.

Zero Extension

- If N is an unsigned binary number, then it is always extended by leading 0s, independent of the value of s.

- Example: N = 10101 01010101

- $N_{zero\text{-}extened}$ =

0000000000000000001010101010101

n-m = 19 13

Address Extension

- Suppose memory address is n-bit, we have to construct it from a short m-bit address field, where n > m.
- Zero extension does not allow m-bit address to refer to all 2^n possible addresses.
- The solution used is to treat the short memory address as a modifier. It is added (in zero-extended form) to a full length memory address stored in a CPU register, known as base register.
- This solution is used in both CISC and RISC processor.

Address Extension

- Example: STB Rs, Rd(S2) is equivalent to

  M(Rd+S2) := Rs[24:31]

- The memory address Rd+S2 is called effective address.

Addressing Modes

- The purpose of the address field is to point to the current value V(X) of some operand X used by the instruction. The way of specifying this value is called addressing mode.

- The addressing mode of X affects the following issues:
  1. The speed with which V(X) can be accessed by the CPU.
  2. The ease with which V(X) can be specified and altered.

- Addressing mode:
  1. Immediate Addressing
  2. Direct Addressing
  3. Indirect Addressing

Immediate Addressing

- The value V(X) of the operand X is contained in the address field itself.
- X is called immediate operand.
- X is constant because it is undesirable to modify instruction fields during execution.
- Example:

  A:=99

  - In Intel 8085 series it is MVI A, 99
  - In Motorola 690X0 series it is MOVE #99, D1 (D1=A)

Direct Addressing

- The address field in the instruction specifies the storage location that contains V(X).
- The storage location can be memory address or register.
- V(X) can be varied without modifying the instruction address field.
- Example:

  A:=B

  - In Intel 8085 series it is MOV A, B

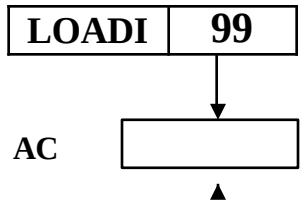
  - In Motorola 690X0 series it is MOVE D2, D1 (D1=A, D2=B)

Indirect Addressing

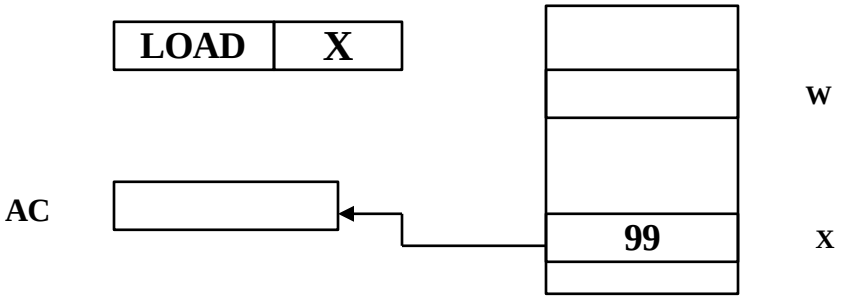
- The instruction contains the address W of a storage location, which in turn contains the address X of the desired operand value V(X).

- By changing the content of W, the address of X can be changed without modifying the instruction address field.

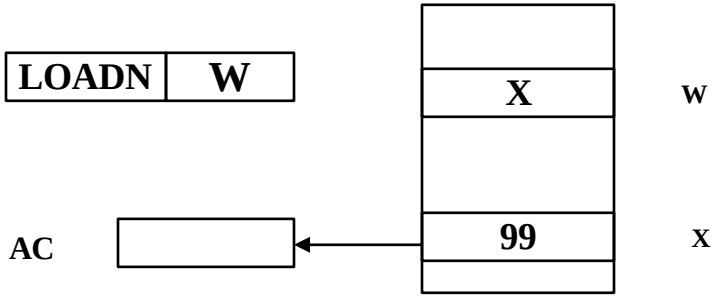
The Three Addressing Modes



Immediate Addressing



Direct Addressing



Indirect Addressing

Relative Addressing

- ***Direct addressing*** requires the complete operand address to appear in the instruction operand field. This address is used without modification. Generally, only partial addressing information is included in the instruction and the CPU must construct the complete (absolute) address.

- ***Relative addressing*** is an address construction technique.

Relative Addressing

- In ***relative addressing***, the operand field contains a relative address, called ***offset/displacement D***. The instruction also implicitly or explicitly identifies other storage locations R_1, R_2,….,R_k (usually CPU registers) containing additional addressing information. The ***effective address A*** of an operand is computed by the function f(D, R_1, R_2,….,R_k). In most cases, ***A:=R+D.***

Advantages of Relative Addressing

- Since all the address information is not included in the instructions, instruction length is reduced.

- By changing the contents of R, the processor can change the absolute addresses referred to by a block of instructions B. It allows the processor to move the entire block B from one region of main memory to another without changing the contents of the instructions of B. Here, R is referred to as a *base register* and its contents as a *base address*.

Advantages of Relative Addressing

- R can be used for storing indexes to facilitate the processing of indexed data. Here, R is called *index register*. The instruction address field D contains the address of the first item X(0) and R contains the index i. The address of item X(i) is D+R. By changing the content of R, a single instruction can be used to access any item X(i) in the given data list.

Drawbacks of Relative Addressing

- Relative addressing requires extra logic circuits and processing time needed to compute addresses.

Auto Indexing

- Indexed items are frequently accessed sequentially. So an access to X(k) stored in memory location A is immediately followed by a reference to X(k+1) or X(k-1) stored in location A+1 or A-1, respectively. So, addressing mode that automatically increment or decrement an address can be defined, which is known as ***auto indexing***.

- Example: **-(A3)** [Motorola 680X0 series] indicate that the content of A3 must be decremented before the instruction is executed. The process is known as ***pre-decrementing***.
  ARM Example: LDR R0, [R1, #4]!

- Similarly, **(A3)+** indicates that A3 should be incremented automatically after the instruction has been executed. This process is called ***post-incrementing.***
  ARM Example: LDR R0, [R1], #4

- In each case the amount of the address increment or decrement is the length in bytes of the indexed operands.

Register Direct Addressing

- In register direct addressing, the name *R* of the register containing the desired value *V(R)* appears in the instruction.

- Example of the Motorola 680X0 instruction:

  MOVE #99, D1

  It means “move the constant 99 to data register D1”. It uses immediate addressing for 99 and register direct addressing for D1.

Register Indirect Addressing

- Register indirect addressing refers to indirect addressing with a register *R* name in the address field.

- Example:

  MOVE.B (A0), D1

  It uses parentheses to indicate that (A0) is an indirect address involving the 680X0's A0 address register.

  This move-byte instruction - the opcode's .B suffix specifies a 1-byte operand – corresponds to

  D1[7:0] := M(A0) and copies the byte addressed by A0 into the low-order byte position of data register D1. The other three bytes of D1 are unchanged.

Register Indirect with Offset

- It is an extension of the register indirect addressing mode.
- It can also be viewed as a type of base or indexed addressing.
- Example:

  SW Rt, OFFSET(Rs)

  corresponds to M(Rs+OFFSET) := Rt,

  where Rs is the base register and OFFSET is a number acting as an offset operand. Rt is also a register.