CPU organizations

- Computers may have instructions of several different lengths containing varying number of addresses.

- The number of address fields in the instruction format of a computer depends on the internal organization of its registers. Most computers fall into one of three types of CPU organizations:

- Single accumulator organization
- General register organization
- Stack organization

Single accumulator organization

- All operations are performed with an implied accumulator register.
- The instruction format in this type of computer uses one address field.
- Ex. – The instruction that specifies an arithmetic addition is defined by an assembly language instruction as

ADD X

Where X is the address of the operand. The ADD instruction in this case results in the operation AC ← AC + M[X].

AC is the accumulator register and M[X] symbolizes the memory word located at address X.

General register organization

- The instruction format in this type of computer needs three register address fields.
- Thus the instruction for an arithmetic addition may be written in an assembly language as

ADD R1, R2, R3

to denote the operation R1 ← R2 + R3.

The number of address fields in the instruction can be reduced from three to two if the destination register is the same as one of the source registers.

General register organization

- Thus the instruction

ADD R1,R2

would denote the operation R1 ← R1 + R2.

- Only register addresses for R1 and R2 need to be specified in this instruction.

General register organization

- General register-type computers employ two or three address fields in their instruction format. Each address field may specify a processor register or memory word. An instruction symbolized by

ADD R1, X

would specify the operation R1 ← R1 + M[X].

It has two address fields, one for the register R1 and other for the memory address X.

Stack organization

- Computers with stack organization would have PUSH and POP instructions which require an address field.
- Thus the instruction

PUSH X

will push the word at address X to the top of the stack. The stack pointer is updated automatically.

- Operation-type instructions do not need an address field in stack-organized computers. This is because the operation is performed on the two items that are on top of the stack.

The instruction

ADD

in a stack computer consists of an operation code only with no address field. This operation has the effect of popping the two top numbers from the stack, adding the numbers, and pushing the sum into the stack.

There is no need to specify operands with an address field since all operands are implied to be in the stack.

Three – Address Instructions

- Computers with three – address instruction formats can use each address field to specify either a processor register or a memory operand.
- The program in assembly language that evaluates
  X = (A+B) * (C+D) is

ADD	R1, A, B	R1 ← M [A] + M [B]
ADD	R2, C, D	R2 ← M [C] + M [D]
MUL	X, R1, R2	M [X] ← R1 * R2

R1 and R2 – Process registers

M[A] - Operand at the memory address symbolized by A.

Three – Address Instructions

- Here, the computer has two process registers, R1 and R2. The symbol M[A] denotes the operand at the memory address symbolized by A.
- The advantage of the three –address format is that it results in short programs when evaluating arithmetic expressions.
- The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two – Address Instructions

- Two address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate

  X = (A+B) * (C+D) is

MOV	R1, A	R1 ← M[A]
ADD	R1, B	R1 ← R1 + M[B]
MOV	R2, C	R2 ← M[C]
ADD	R2, D	R2 ← R2 + M[D]
MUL	R1, R2	R1 ← R1 * R2
MOV	X, R1	M[X] ← R1

Two – Address Instructions

- The MOV instruction moves or transfers the operands to and from memory and a processor registers.

- The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

One – Address Instructions

- One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register.

- However, here we will neglect the second register and assume that the AC contains the result of all operations.

One – Address Instructions

- The program to evaluate X=(A+B)*(C+D) is

```
LOAD     A     AC ← M[A]
ADD      B     AC ← AC + M[B]
STORE    T     M[T] ← AC
LOAD     C     AC ← M[C]
ADD      D     AC ← AC + M[D]
MUL      T     AC ← AC * M[T]
STORE    X     M[X] ← AC
```

Zero – Address Instructions

- A stack-organized computer does not use an address field for the instructions ADD and MUL.

  The PUSH and POP instructions, however need an address field to specify the operand that communicates with the stack.

  The following program shows how X = (A+B)*(C+D) will be written for a stack- organized computer.

- TOS stands for Top Of the Stack.

Zero – Address Instructions

PUSH	A	TOS ← A
PUSH	B	TOS ← B
ADD		TOS ← (A + B)
PUSH	C	TOS ← C
PUSH	D	TOS ← D
ADD		TOS ← (C + D)
MUL		TOS ← (C + D) * (A + B)
POP	X	M[X] ← TOS

Zero – Address Instructions

- All operations are done between the AC register and a memory operand. T is the address of a temporary memory location required for storing the intermediate result.

CISC Instructions

- CISC stands for Complex Instruction Set Computer

- The design of an instruction set for a computer must take into consideration not only machine language constructs, but also the requirements imposed on the use of high-level programming languages.

CISC Instructions

The major characteristics of CISC architecture are:

- 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.
- Variable – length instruction formats
- Instructions that manipulate operands in memory.

RISC Instructions

- RISC – Reduced Instruction Set Computer

- The concept of RISC architecture involves an attempt to reduce execution time by simplifying the instruction set of the computer. The major characteristics of a RISC processor are:

- Relatively few instructions

RISC Instructions

- Relative few addressing modes
- Memory access limited to load and store instructions.
- All operations done within the registers of the CPU
- Fixed-length, easily decoded instruction format.
- Single – cycle instruction execution.
- Hardwired rather than microprogrammed control.

Addressing modes