Instruction Set Architecture (Continued)

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 $AC \leftarrow AC + M[X]$.

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 $R1 \leftarrow R2 + R3$.

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 $R1 \leftarrow R1 + R2$.

- 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 $R1 \leftarrow R1 + M[X]$.

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.

Stack organization

The instruction

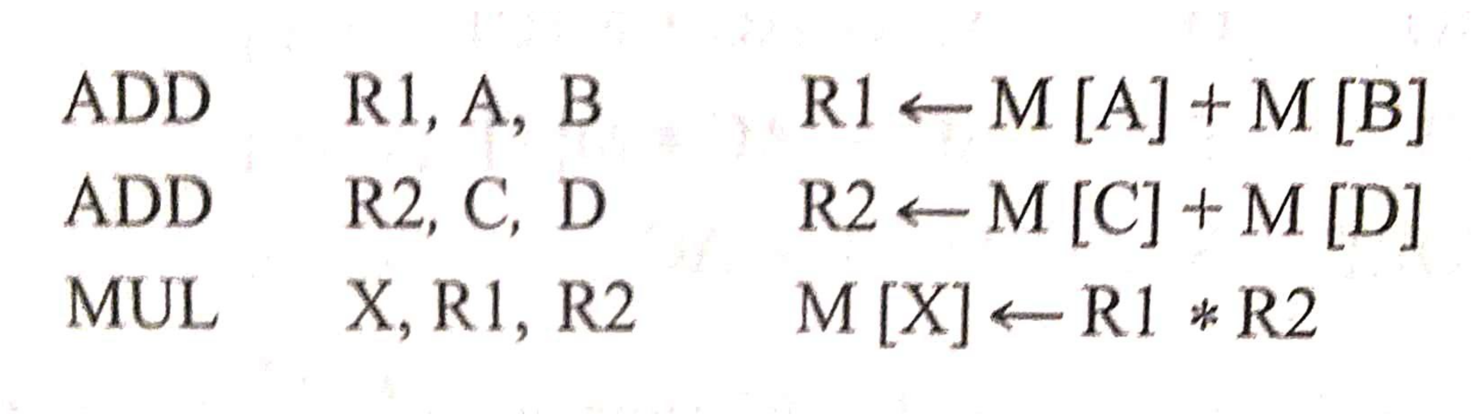
ADD

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.

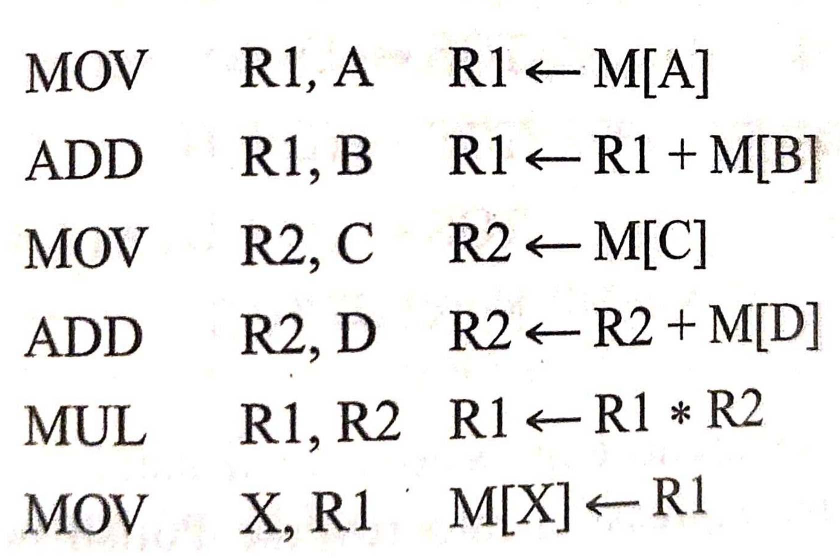
- The advantage of the three –address format is that it results in short programs when evaluating arithmetic expressions.

- 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.

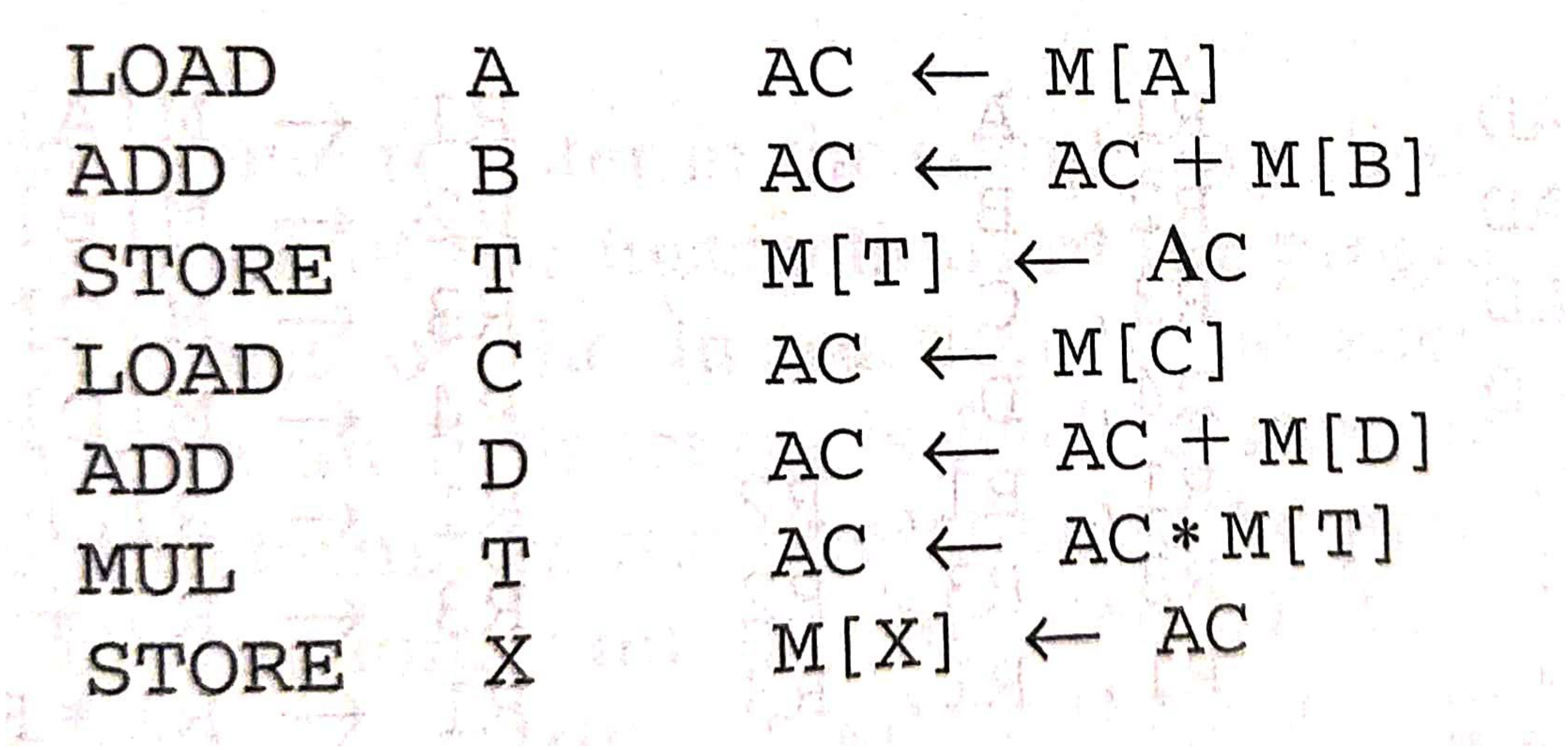
One – Address Instructions

- 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

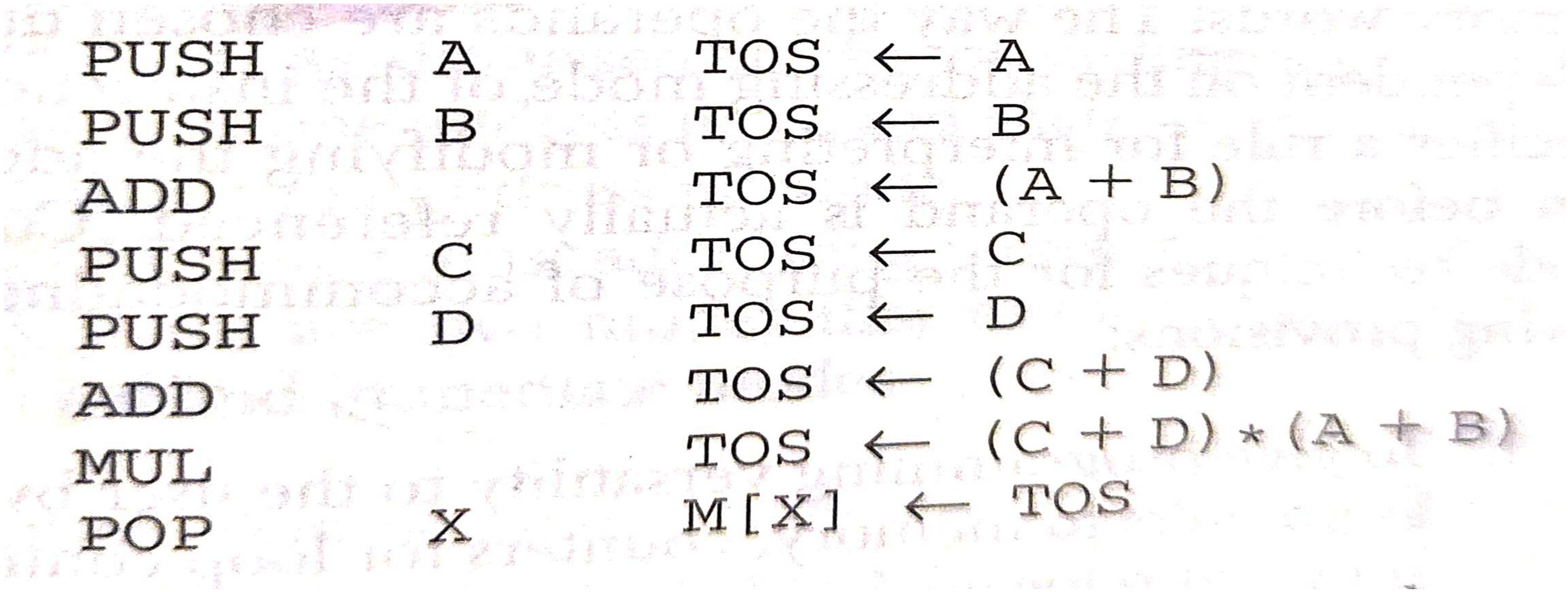
- A stack-organized computer does not use an address field for the instructions ADD and MUL.

  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

What is an instruction?

Format of an instruction

Mode	**opcode**	**operand**

What is an instruction?

Mode	**opcode**	**operand**

Mode means addressing mode i.e to find out the address of the data that we are searching. This data may be in the memory or in a register.

Addressing mode tells us the address of memory or register.

What is an instruction?

Mode	opcode	operand

- opcode means operation – addition, subtraction, etc.

- Operand may be data, sometimes operand may be address of any memory location, sometimes operand may be address of any register. It can also be any memory location which contains address of another memory location.

Addressing modes

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer registers or memory words.

The way the operands are chosen during program execution is dependent on the addressing mode of the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Addressing modes

Program counter

Implied mode

Immediate mode

Register mode

Register Indirect mode

Addressing modes

Autoincrement or Autodecrement Mode

Direct Address Mode

Indirect Address Mode

Relative Address Mode

Indexed Addressing

- Base Register Addressing Mode

Program counter

- There is one register in the computer called the program counter or PC that keeps track of the instructions in the program stored in the memory.

- PC holds the address of the instruction to be executed next and is incremented each time an instruction is fetched from the memory.

Implied Mode

- In this mode, the operands are specified implicitly in the definition of the instruction.
- For example, the instruction “ complement accumulator’ is an implied – mode instruction because the operand in the accumulator register is implied in the definition of the instruction.
- In fact, all register reference instruction that use an accumulator are implied – mode instructions.

Immediate Mode

In this mode, the operand is specified in the instruction itself.

In other words, an immediate – mode instruction has an operand field rather than an address field.

The operand field contains the actual to be used in conjunction with the operation specified in the instruction.

Utility

Immediate – mode instructions are useful for initializing registers to a constant value.

Register mode

In this mode, the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2^k registers.

Register Indirect mode

In this mode, the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

In other words, the selected register contains the address of the operand rather than the operand itself.

Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.

Register Indirect mode

- The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

Autoincrement or Autodecrement Mode

- This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.
- When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be achieved by using the increment or decrement instruction.

Direct Address Mode

- In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction.

- In a branch – type instruction, the address field specifies the actual branch address.

Indirect Address mode

- In this mode, the address field of the instruction gives the address where the effective address is stored in memory.

- Control fetches the instruction from the memory and uses its address part to access memory again to read the effective address.

- A few addressing modes require that the address field of the instruction be added to the content of a specific register in the CPU

Indirect Address mode

- Effective address in these modes is obtained from the following computation:

- Effective address = address part of instruction + content of CPU register.

Relative Address Mode

- In this mode, the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually a signed number (in 2’s complement representation) which can be either positive or negative.

Indexed Addressing Mode

- In this mode, the content of an index register is added to the address part of the instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value. The address field of the instruction defines the beginning address of a data array in memory.
- Each operand in the array is stored in memory relative to the begining address.

Indexed Addressing Mode

- The distance between the beginning address and the address of the operand is the index value stored in the index register.
- Any operand in the array can be accessed with the same instruction provided that the index register contains the correct index value.
- The index register can be incremented to facilitate access to consecutive operands.

Base Register Addressing Mode

- In this mode, the content of a base register is added to the address part of the instruction to obtain the effective address.

- This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register.

- The difference between the two modes is in the way they are used rather than in the way that they are computed.

Base Register Addressing Mode

- An index register is assumed to hold an index number that is relative to the address part of the instruction.
- A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.
- The base register addressing mode is used in computers to facilitate the relocation of the programs in memory.

Thank you