

Characteristic of RISC –

1. Simpler instruction, hence simple instruction decoding.
2. Instruction come under size of one word.
3. Instruction take single clock cycle to get executed.
4. More number of general purpose register.
5. Simple Addressing Modes.
6. Less Data types.
7. Pipeline can be achieved.

Characteristic of CISC –

1. Complex instruction, hence complex instruction decoding.
2. Instruction are larger than one word size.
3. Instruction may take more than single clock cycle to get executed.
4. Less number of general purpose register as operation get performed in memory itself.
5. Complex Addressing Modes.
6. More Data types.

Example – Suppose we have to add two 8-bit number:

- **CISC approach:** There will be a single command or instruction for this like ADD which will perform the task.
- **RISC approach:** Here programmer will write first load command to load data in registers then it will use suitable operator and then it will store result in desired location.

So, add operation is divided into parts i.e. load, operate, store due to which RISC programs are longer and require more memory to get stored but require less transistors due to less complex command.

Difference –

RISC

Focus on software

Uses only Hardwired control unit

CISC

Focus on hardware

Uses both hardwired and micro programmed control unit

RISC

Transistors are used for more registers

Fixed sized instructions

Can perform only Register to Register Arithmetic operations

Requires more number of registers

Code size is large

A instruction execute in single clock cycle

A instruction fit in one word

CISC

Transistors are used for storing complex Instructions

Variable sized instructions

Can perform REG to REG or REG to MEM or MEM to MEM

Requires less number of registers

Code size is small

Instruction take more than one clock cycle

Instruction are larger than size of one word

Arithmetic Instructions

Instructions of this group perform addition, subtraction, multiplication, division, increment, decrement, comparison, ASCII and decimal adjustment etc.

The following instructions come under this category:

Instruction	Description
ADD	Adds data to the accumulator i.e. AL or AX register or memory locations.
ADC	Adds specified operands and the carry status (i.e. carry of the previous stage).
SUB	Subtract immediate data from accumulator, memory or register.
SBB	Subtract immediate data with borrow from accumulator, memory or register.
MUL	Unsigned 8-bit or 16-bit multiplication.
IMUL	Signed 8-bit or 16-bit multiplication.
DIV	Unsigned 8-bit or 16-bit division.
IDIV	Signed 8-bit or 16-bit division.
INC	Increment Register or memory by 1.
DEC	Decrement register or memory by 1.
DAA	Decimal Adjust after BCD Addition: When two BCD numbers are added, the DAA is used after ADD or ADC instruction to get correct answer in BCD.
DAS	Decimal Adjust after BCD Subtraction: When two BCD numbers are added, the DAS is used after SUB or SBB instruction to get correct answer in BCD.

AAA	ASCII Adjust for Addition: When ASCII codes of two decimal digits are added, the AAA is used after addition to get correct answer in unpacked BCD.
AAD	Adjust AX Register for Division: It converts two unpacked BCD digits in AX to the equivalent binary number. This adjustment is done before dividing two unpacked BCD digits in AX by an unpacked BCD byte.
AAM	Adjust result of BCD Multiplication: This instruction is used after the multiplication of two unpacked BCD.
AAS	ASCII Adjust for Subtraction: This instruction is used to get the correct result in unpacked BCD after the subtraction of the ASCII code of a number from ASCII code another number.
CBW	Convert signed Byte to signed Word.
CWD	Convert signed Word to signed Doubleword.
NEG	Obtains 2's complement (i.e. negative) of the content of an 8-bit or 16-bit specified register or memory location(s).
CMP	Compare Immediate data, register or memory with accumulator, register or memory location(s).

Logical Instructions

Instruction of this group perform logical AND, OR, XOR, NOT and TEST operations. **The following instructions come under this category:**

Instruction	Description
AND	Performs bit by bit logical AND operation of two operands and places the result in the specified destination.
OR	Performs bit by bit logical OR operation of two operands and places the result in the specified destination.
XOR	Performs bit by bit logical XOR operation of two operands and places the result in the specified destination.
NOT	Takes one's complement of the content of a specified register or memory location(s).
TEST	Perform logical AND operation of a specified operand with another specified operand.

Rotate Instructions

The following instructions come under this category:

Instruction	Description
RCL	Rotate all bits of the operand left by specified number of bits through carry flag.
RCR	Rotate all bits of the operand right by specified number of bits through carry flag.

ROL	Rotate all bits of the operand left by specified number of bits.
ROR	Rotate all bits of the operand right by specified number of bits.

Shift Instructions

The following instructions come under this category:

Instruction	Description
SAL or SHL	Shifts each bit of operand left by specified number of bits and put zero in LSB position.
SAR	Shift each bit of any operand right by specified number of bits. Copy old MSB into new MSB.
SHR	Shift each bit of operand right by specified number of bits and put zero in MSB position.

Branch Instructions

It is also called program execution transfer instruction. Instructions of this group transfer program execution from the normal sequence of instructions to the specified destination or target. The following instructions come under this category:

Instruction	Description
JA or JNBE	Jump if above, not below, or equal i.e. when CF and ZF = 0

JAE/JNB/JNC	Jump if above, not below, equal or no carry i.e. when $CF = 0$
JB/JNAE/JC	Jump if below, not above, equal or carry i.e. when $CF = 0$
JBE/JNA	Jump if below, not above, or equal i.e. when CF and $ZF = 1$
JCXZ	Jump if CX register = 0
JE/JZ	Jump if zero or equal i.e. when $ZF = 1$
JG/JNLE	Jump if greater, not less or equal i.e. when $ZF = 0$ and $CF = OF$
JGE/JNL	Jump if greater, not less or equal i.e. when $SF = OF$
JL/JNGE	Jump if less, not greater than or equal i.e. when $SF \neq OF$
JLE/JNG	Jump if less, equal or not greater i.e. when $ZF = 1$ and $SF \neq OF$
JMP	Causes the program execution to jump unconditionally to the memory address or label given in the instruction.
CALL	Calls a procedure whose address is given in the instruction and saves their return address to the stack.
RET	Returns program execution from a procedure (subroutine) to the next instruction or main program.

IRET	Returns program execution from an interrupt service procedure (subroutine) to the main program.
INT	Used to generate software interrupt at the desired point in a program.
INTO	Software interrupts to indicate overflow after arithmetic operation.
LOOP	Jump to defined label until CX = 0.
LOOPZ/LOOPE	Decrement CX register and jump if CX ≠ 0 and ZF = 1.
LOOPNZ/LOOPNE	Decrement CX register and jump if CX ≠ 0 and ZF = 0.

Here, CF = Carry Flag
ZF = Zero Flag
OF = Overflow Flag
SF = Sign Flag
CX = Register

Flag Manipulation and Processor Control Instructions

Instructions of this instruction set are related to flag manipulation and machine control. The following instructions come under this category:

Instruction	Description
CLC	Clear Carry Flag: This instruction resets the carry flag CF to 0.

CLD	Clear Direction Flag: This instruction resets the direction flag DF to 0.
CLI	Clear Interrupt Flag: This instruction resets the interrupt flag IF to 0.
CMC	This instruction take complement of carry flag CF.
STC	Set carry flag CF to 1.
STD	Set direction flag to 1.
STI	Set interrupt flag IF to 1.
HLT	Halt processing. It stops program execution.
NOP	Performs no operation.
ESC	Escape: makes bus free for external master like a coprocessor or peripheral device.
WAIT	When WAIT instruction is executed, the processor enters an idle state in which the processor does no processing.
LOCK	It is a prefix instruction. It makes the LOCK pin low till the execution of the next instruction.

String Instructions

String is series of bytes or series of words stored in sequential memory locations. The 8086 provides some instructions which handle string operations such as string movement, comparison, scan, load and store.

The following instructions come under this category:

Instruction	Description
MOVS/MOVS _B /MOVS _W	Moves 8-bit or 16-bit data from the memory location(s) addressed by SI register to the memory location addressed by DI register.
CMPS/CMPS _B /CMPS _W	Compares the content of memory location addressed by DI register with the content of memory location addressed by SI register.
SCAS/SCAS _B /SCAS _W	Compares the content of accumulator with the content of memory location addressed by DI register in the extra segment ES.
LODS/LODS _B /LODS _W	Loads 8-bit or 16-bit data from memory location addressed by SI register into AL or AX register.
STOS/STOS _B /STOS _W	Stores 8-bit or 16-bit data from AL or AX register in the memory location addressed by DI register.
REP	Repeats the given instruction until CX ≠ 0
REPE/ REPZ	Repeats the given instruction till CX ≠ 0 and ZF = 1
REPNE/REPNZ	Repeats the given instruction till CX ≠ 0 and ZF = 0