INTROCTION ABOUT CPU, REGISTERS AND MEMORY, GENERAL REGISTER ORGANIZATION

A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters). Some instructions specify registers as part of the instruction. For example, an instruction may specify that the contents of two defined registers be added together and then placed in a specified register.

Registers are the most important components of CPU. Each register performs a specific function. A brief description of most important CPU's registers and their functions are given below:

1. **Memory Address Register (MAR):** This register holds the address of memory where CPU wants to read or write data. When CPU wants to store some data in the memory or reads the data from the memory, it places the address of the required memory location in the MAR.

2. **Memory Buffer Register (MBR):** This register holds the contents of data or instruction read from, or written in memory. The contents of instruction placed in this register are transferred to the Instruction Register, while the contents of data are transferred to the accumulator or I/O register. In other words you can say that this register is used to store data/instruction coming from the memory or going to the memory.

3. **Program Counter (PC):** Program Counter register is also known as Instruction Pointer Register. This register is used to store the address of the next instruction to be fetched for execution. When the instruction is fetched, the value of IP is incremented. Thus this register always points or holds the address of next instruction to be fetched.

4. **Instruction Register (IR):** Once an instruction is fetched from main memory, it is stored in the Instruction Register. The control unit takes instruction from this register, decodes and executes it by sending signals to the appropriate component of computer to carry out the task.

5. **Accumulator Register:** The accumulator register is located inside the ALU; it is used during arithmetic & logical operations of ALU. The control unit stores data values fetched from main memory in the accumulator for arithmetic or logical operation. This register holds the initial data to be operated upon, the intermediate results, and the final result of operation. The final result is transferred to main memory through MBR.

6. **Stack Control Register:** A stack represents a set of memory blocks; the data is stored in and retrieved from these blocks in an order, i.e. First In and Last Out (FILO). The Stack Control Register is used to manage the stacks in memory. The size of this register is 2 or 4 bytes.

7. **Flag Register:** The Flag register is used to indicate occurrence of a certain condition during an operation of the CPU. It is a special purpose register with size one byte or two bytes. Each bit of the flag register constitutes a flag (or alarm), such that the bit value indicates if a specified condition was encountered while executing an instruction.

For example, if zero value is put into an arithmetic register (accumulator) as a result of an arithmetic operation or a comparison, then the zero flag will be raised by the CPU. Thus, the subsequent instruction can check this flag and when a zero flag is "ON" it can take an appropriate route in the algorithm.

Memory

This unit can store instructions, data, and intermediate results. This unit supplies information to other units of the computer when needed. It is also known as internal storage unit or the main memory or the primary storage or Random Access Memory (RAM).

Its size affects speed, power, and capability. Primary memory and secondary memory are two types of memories in the computer. Functions of the memory unit are −

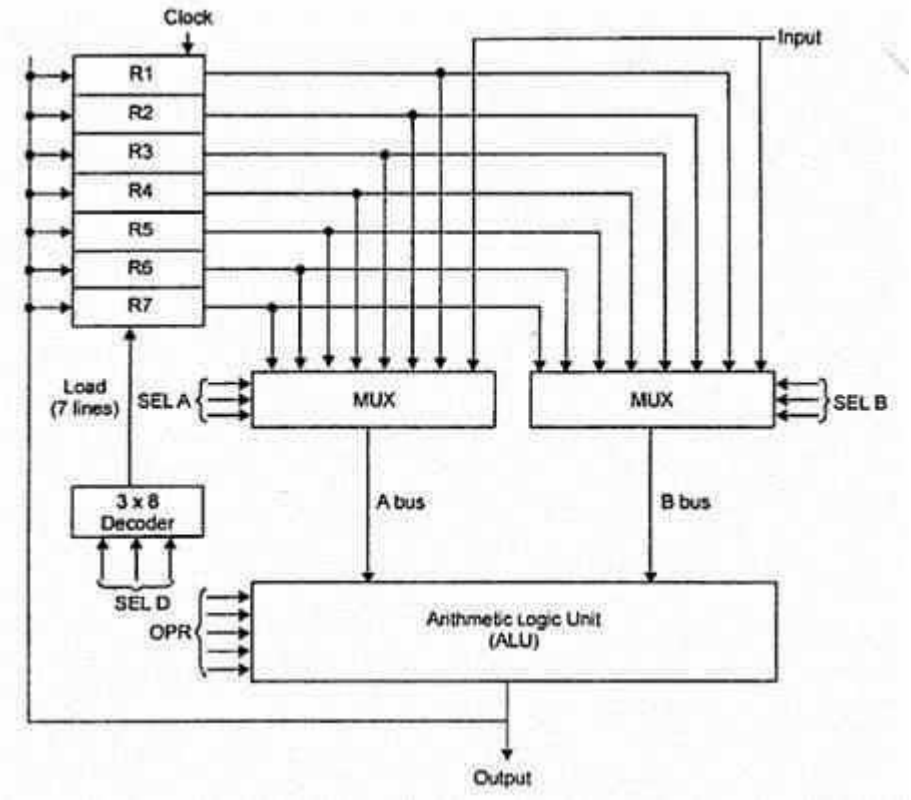
1. It stores all the data and the instructions required for processing.
2. It stores intermediate results of processing.
3. It stores the final results of processing before these results are released to an output device.
4. All inputs and outputs are transmitted through the main memory.

General Register organization

Generally CPU has seven general registers. Register organization show how registers are selected and how data flow between register and ALU. A decoder is used to select a

particular register. The output of each register is connected to two multiplexers to form the two buses A and B. The selection lines in each multiplexer select the input data for the particular bus.

The A and B buses form the two inputs of an ALU. The operation select lines decide the micro operation to be performed by ALU. The result of the micro operation is available at the output bus. The output bus connected to the inputs of all registers, thus by selecting a destination register it is possible to store the result in it.



A bus organization for seven CPU registers

EXAMPLE:

- To perform the operation ***R3 = R1+R2*** **we** have to provide following binary selection variable to the select inputs.

1. **SEL A**: **001** -To place the contents of R1 into bus A.
2. **SEL B**: **010** - to place the contents of R2 into bus B
3. **SEL OPR**: **10010** – to perform the arithmetic addition A+B
4. **SEL REG or SEL D**: **011** – to place the result available on output bus in R3.

Register and multiplexer input selection code

Binary code	SELA	SELB	SELD or SELREG
000	Input	Input	---
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Operation with symbol

Operation selection code	Operation	symbol
0000	Transfer A	TSFA
0001	Increment A	INC A
0010	A+B	ADD
0011	A-B	SUB
0100	Decrement A	DEC
0101	A AND B	AND
0110	A OR B	OR
0111	A XOR B	XOR
1000	Complement A	COMA
1001	Shift right A	SHR
1010	Shift left A	SHL

What is CONTROL WORD?

- The combined value of a binary selection inputs specifies the control word.
- It consists of four fields SELA, SELB, and SELD or SELREG contains three bit each and SELOPR field contains four bits thus the total bits in the control word are 13-bits.

SEL A	SELB	SELREG OR SELD	SELOPR

FORMATE OF CONTROL WORD

1. The three bit of SELA select a source registers of the input of the ALU.
2. The three bits of SELB select a source registers of the b input of the ALU.
3. The three bits of SELED or SELREG select a destination register using the decoder.
4. The four bits of SELOPR select the operation to be performed by ALU.

CONTROL WORD FOR OPERATION R2 = R1+R3

SEL A	SEL B	SEL D OR SELREG	SELOPR
001	011	010	0010

Note: Control words for all micro operation are stored in the control memory

Example:

MICROOPERATION	SEL A	SEL B	SEL D OR SELREG	SELOPR	CONTROL WORD			
R2 = R1+R3	R1	R3	R2	ADD	00 1	01 1	01 0	001 0

DAY 3-10 STACK ORGANIZATION, INSTRUCTION FORMAT &ADDRESSING MODE

STACK ORGANIZATION

Stack is a storage structure that stores information in such a way that the last item stored is the first item retrieved. It is based on the principle of LIFO (Last-in-first-out). The stack in digital computers is a group of memory locations with a register that holds the address of top of element. This register that holds the address of top of element of the stack is called ***Stack Pointer***.

Stack Operations

The two operations of a stack are:

1. **Push:** Inserts an item on top of stack.
2. **Pop:** Deletes an item from top of stack.

Implementation of Stack

In digital computers, stack can be implemented in two ways:

1. Register Stack
2. Memory Stack

Register Stack

A stack can be organized as a collection of finite number of registers that are used to store temporary information during the execution of a program. The stack pointer (SP) is a register that holds the address of top of element of the stack.

Memory Stack

A stack can be implemented in a random access memory (RAM) attached to a CPU. The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer. The starting memory location of the stack is specified by the processor register as *stack pointer*.

INSTRUCTION FORMATS

The physical and logical structure of computers is normally described in reference manuals provided with the system. Such manuals explain the internal construction of the CPU, including the processor registers available and their logical capabilities. They list all

hardware-implemented instructions, specify their binary code format, and provide a precise definition of each instruction. A computer will usually have a variety of instruction code formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction.

The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register. The bits of the instruction are divided into groups called fields. The most common fields found in instruction formats are:

1. An operation code field that specifies the operation to be performed.
2. An address field that designates a memory address or a processor registers.
3. A mode field that specifies the way the operand or the effective address is determined.

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:

1 Single accumulator organization.
2 General register organization.
3 Stack organization.

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 shown below, together with comments that explain the register transfer operation of each instruction.

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

It is assumed that the computer has two processor registers, R1 and R2. The symbol M [A] denotes the operand at 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. An example of a commercial computer that uses three-address instructions is the Cyber 170. The instruction formats in the Cyber computer are restricted to either three register address fields or two register address fields and one memory address field.

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 as follows:

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

The MOV instruction moves or transfers the operands to and from memory and 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 and assume that the AC contains the result of tall operations. The program to evaluate X = (A + B) ∗ (C + D) is

LOAD A	AC ← M [A]
ADD B	AC ← A [C] + 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

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.

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 stack)

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

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions.

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 in dependent on the addressing mode of the instruction. 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.

Although most addressing modes modify the address field of the instruction, there are two modes that need no address field at all. These are the implied and immediate modes.

1 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 instructions that use an accumulator are implied-mode instructions. Instruction format with mode field Zero-address instructions in a stack-

organized computer are implied-mode instructions since the operands are implied to be on top of the stack.

2 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 operand to be used in conjunction with the operation specified in the instruction. Immediate-mode instructions are useful for initializing registers to a constant value. It was mentioned that the address field of an instruction may specify either a memory word or a processor register. When the address field specifies a processor register, the instruction is said to be in the register mode.

3 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 2k registers.

4 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 Op code Mode Address operand itself. Before using a register indirect mode instruction, the programmer must ensure that the memory address for the operand is placed in the processor register with a previous instruction. A reference to the register is then equivalent to specifying a memory address. 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.

5 Auto increment or Auto decrement 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. However, because it is such a common requirement, some computers incorporate a special mode that automatically increments or decrements the content of the register after data access. The address field of an instruction is used by the control unit in the CPU to obtain the operand from memory. Sometimes the value given in the address field is the address of the operand, but sometimes it

is just an address from which the address of the operand is calculated. To differentiate among the various addressing modes it is necessary to distinguish between the address part of the instruction and the effective address used by the control when executing the instruction. The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational-type instruction. It is the address where control branches in response to a branch-type instruction.

6 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.

7 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 memory and uses its address part to access memory again to read the effective address.

8 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. When this number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction. To clarify with an example, assume that the program counter contains the number 825 and the address part of the instruction contains the number 24. The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826 + 24 = 850. This is 24 memory locations forward from the address of the next instruction. Relative addressing is often used with branch-type instructions when the branch address is in the area surrounding the instruction word itself. It results in a shorter address field in the instruction format since the relative address can be specified with a smaller number of bits compared to the number of bits required to designate the entire memory address.

9 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 beginning address. The distance between the beginning address and the address of the operand is the index value stores 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. Note that if an index-type instruction does not include an address field in its format, the instruction converts to the register indirect mode of operation. Some computers dedicate one CPU register to function solely as an index register. This register is involved implicitly when the index-mode instruction is used. In computers with many processor registers, any one of the CPU registers can contain the index number. In such a case the register must be specified explicitly in a register field within the instruction format.

10 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. 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 programs in memory. When programs and data are moved from one segment of memory to another, as required in multiprogramming systems, the address values of the base register requires updating to reflect the beginning of a new memory segment.

CONCEPT OF CPU DESIGN, RISC & CISC

Day 12 to 18

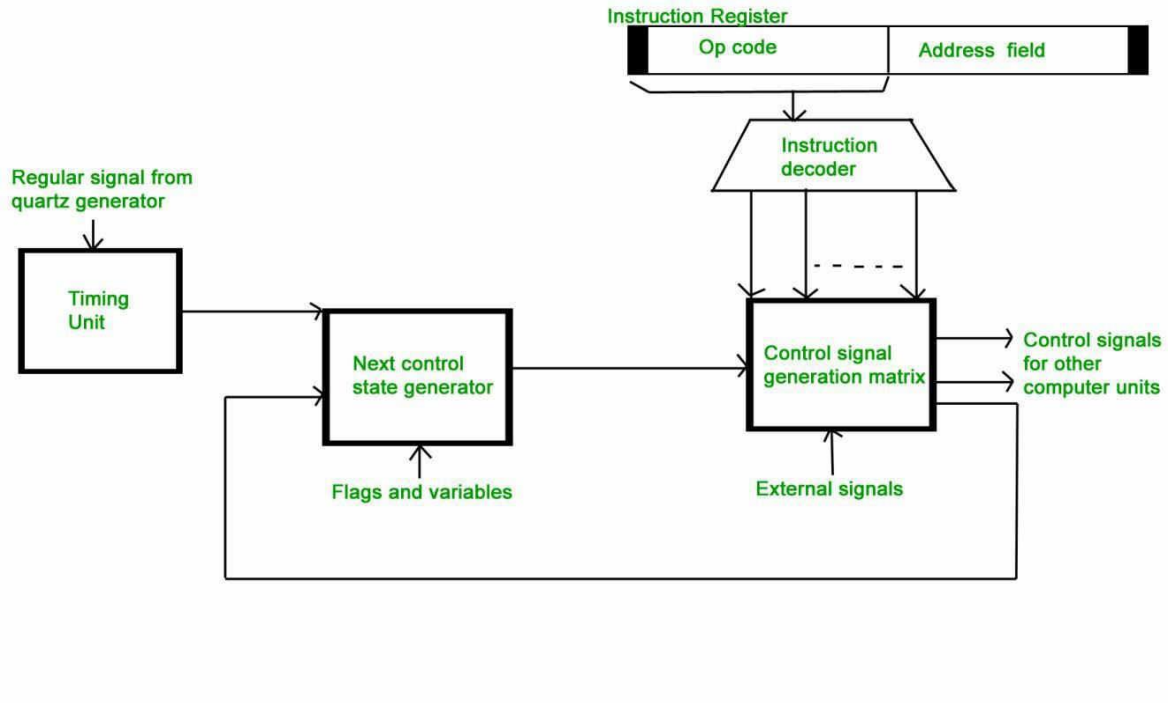
Hardwired v/s Micro-programmed Control Unit

To execute an instruction, the control unit of the CPU must generate the required control signal in the proper sequence. There are two approaches used for generating the control signals in proper sequence as Hardwired Control unit and Micro-programmed control unit.

Hardwired Control Unit –

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, the condition codes and the external inputs. The outputs of the state machine are the control signals. The sequence of the operation carried out by this machine is determined by the wiring of the logic elements and hence named as "hardwired".

1. Fixed logic circuits that correspond directly to the Boolean expressions are used to generate the control signals.
2. Hardwired control is faster than micro-programmed control.
3. A controller that uses this approach can operate at high speed.

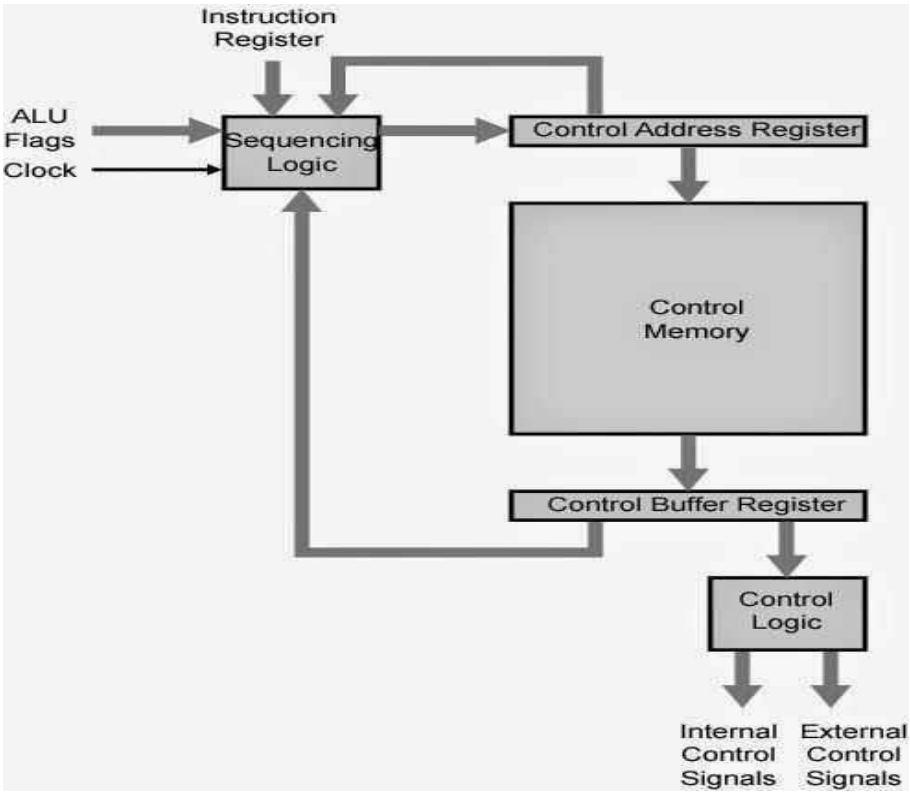


Characteristics:

1. It uses flags, decoder, logic gates and other digital circuits.
2. As name implies it is a hardware control unit.
3. On the basis of input Signal output is generated.
4. Difficult to design, test and implement.
6. Inflexible to modify.
7. Faster mode of operation.
8. Expensive and high error.
9. Used in RISC processor.

Micro-programmed Control Unit –

1. The control signals associated with operations are stored in special memory units inaccessible by the programmer as Control Words.
2. Control signals are generated by a program are similar to machine language programs.
3. Micro-programmed control unit is slower in speed because of the time it takes to fetch microinstructions from the control memory.



Characteristics

1. It uses sequence of micro-instruction in micro programming language.

2. It is mid-way between Hardware and Software.

3. It generates a set of control signal on the basis of control line.

4. Easy to design, test and implement.

5. Flexible to modify.

6. Slower mode of operation.

7. Cheaper and less error.

8. Used in CISC processor.

Reduced Instruction Set Computer

A reduced instruction set computer (RISC) is a computer that uses a central processing unit (CPU) that implements the processor design principle of simplified instructions. To date, RISC is the most efficient CPU architecture technology.

This architecture is an evolution and alternative to complex instruction set computing (CISC). With RISC, the basic concept is to have simple instructions that do less but execute very quickly to provide better performance.

The most basic RISC feature is a processor with a small core logic that allows engineers to increase the register set and increase internal parallelism by using the following:

1. Thread level parallelism: Increases the number of parallel threads executed by the CPU
2. Instruction level parallelism: Increases the speed of the CPU's executing instructions

The words "reduced instruction set" are often misinterpreted to refer to a reduced number of instructions. However, this is not the case, as several RISC processors, like the PowerPC, have numerous instructions. At the opposite end of the spectrum, the DEC PDP-8, a CISC CPU, has only eight basic instructions. Reduced instruction actually means that the amount of work done by each instruction is reduced in terms of number of cycles - at most only a single data memory cycle - compared to CISC CPUs, in which dozens of cycles are required prior to completing the entire instruction. This results in faster processing.

CISC

The main intend of the CISC processor architecture is to complete task by using less number of assembly lines. For this purpose, the processor is built to execute a series of operations. Complex instruction is also termed as MULT, which operates memory banks of a computer directly without making the compiler to perform storing and loading functions.

Features of CISC Architecture

1. To simplify the computer architecture, CISC supports microprogramming.
2. CISC have more number of predefined instructions which makes high level languages easy to design and implement.
3. CISC consists of less number of registers and more number of addressing modes, generally 5 to 20.
4. CISC processor takes varying cycle time for execution of instructions – multi-clock cycles.
5. Because of the complex instruction set of the CISC, the pipelining technique is very difficult.

6. CISC consists of more number of instructions, generally from 100 to 250.
7. Special instructions are used very rarely.
8. Operands in memory are manipulated by instructions.

Advantages of CISC architecture

1. Each machine language instruction is grouped into a microcode instruction and executed accordingly, and then are stored inbuilt in the memory of the main processor, termed as microcode implementation.
2. As the microcode memory is faster than the main memory, the microcode instruction set can be implemented without considerable speed reduction over hard wired implementation.
3. Entire new instruction set can be handled by modifying the micro program design.
4. CISC, the number of instructions required to implement a program can be reduced by building rich instruction sets and can also be made to use slow main memory more efficiently.
5. Because of the superset of instructions that consists of all earlier instructions, this makes micro coding easy.

Drawbacks of CISC

1. The amount of clock time taken by different instructions will be different – due to this – the performance of the machine slows down.
2. The instruction set complexity and the chip hardware increases as every new version of the processor consists of a subset of earlier generations.
3. Only 20% of the existing instructions are used in a typical programming event, even though there are many specialized instructions in existence which are not even used frequently.
4. The conditional codes are set by the CISC instructions as a side effect of each instruction which takes time for this setting – and, as the subsequent instruction changes the condition code bits – so, the compiler has to examine the condition code bits before this happens.

Difference between CISC and RISC

Architectural Characteristics	Complex Instruction Set Computer(CISC)	Reduced Instruction Set Computer(RISC)
Instruction size and format	Large set of instructions with variable formats (16-64 bits per instruction).	Small set of instructions with fixed format (32 bit).
Data transfer	Memory to memory.	Register to register.
CPU control	Most micro coded using control memory (ROM) but modern CISC use hardwired control.	Mostly hardwired without control memory.
Instruction type	Not register based instructions.	Register based instructions.
Memory access	More memory access.	Less memory access.
Clocks	Includes multi-clocks.	Includes single clock.
Instruction nature	Instructions are complex.	Instructions are simple.