Comparison between risc and cisc
Anurag parmar , Deepak Kumar, Indu bala

Abstract: RISC vs. CISC wars raged in the 1980s when chip area and processor design complexity were the primary constraints and desktops and servers exclusively dominated the computing landscape. The main difference between RISC and CISC is in the number of computing cycles each of their instructions take. The difference the number of cycles is based on the complexity and the goal of their instructions. The main goal of this paper is to compare the RISC and CISC On the basis of the various terms. In this paper we are describing the RISC and CISC (reduced instructions set computer and complex instruction set computer resp.) using the two microcontroller that are 8051 and PIC.

INTRODUCTION:
RISC and CISC stand for two different competing philosophies in designing modern computer architecture. The debate between them has been going on for a long time and will likely continue. The difference between RISC and CISC can lays on many levels, lots of plausible arguments are put forward by both sides, such as code density, transistor counts, memory compiler and decode complexity etc. RISC and CISC are two important philosophies in designing the computer architecture these two terms are compared with each other in various terms that are cycles, program, design, complexity, memory unit and pipelining etc.

RISC:
The term RISC stands for ‘Reduced Instruction Set Computer’. It is a CPU design strategy based on simple instructions and fast performance. RISC is small or reduced set of instructions. Here, each instruction is meant to achieve very small tasks. In a RISC machine, the instruction sets are simple and basic, which help in composing more complex instructions. Each instruction is of the same length; the instructions are strung together to get complex tasks done in a single operation. Most instructions are completed in one machine cycle. This pipelining is a key technique used to speed up RISC machines.

RISC is a microprocessor that is designed to carry out few instructions at the same time. Based on small instructions, these chips require fewer transistors, which make the transistors cheaper to design and produce. Some other features of RISC include:
• Less decoding demand
• Uniform instruction set
• Identical general purpose registers
• Simple addressing nodes
• Few data types in hardware

Also, while writing codes, RISC makes it easier by allowing the programmer to remove unnecessary codes and prevents wasting of cycles.

History of risc:
The first RISC projects came from IBM, Stanford, and UC-Berkeley in the late 70s and early 80s. The IBM 801, Stanford MIPS, and Berkeley RISC 1 and 2 were all designed with a similar philosophy which has become known as RISC. Certain design features have been characteristic of most RISC processors:

one cycle execution time: RISC processors have a CPI (clock per instruction) of one cycle. This is due to the optimization of each instruction on the CPU and a technique called PIPELINING

pipelining: a technique that allows for simultaneous execution of parts, or stages, of instructions to more efficiently process instructions;
Large number of registers: the RISC design philosophy generally incorporates a larger number of registers to prevent in large amounts of interactions with memory.

RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:
- fetch instructions from memory
- read registers and decode the instruction
- execute the instruction or calculate an address
- access an operand in data memory
- write the result into a register

CISC:
The term CISC stands for ‘Complex Instruction Set Computer’. It is a CPU design strategy based on single instructions, which are capable of per CISC computers have shorted programs. It has a large number of complex instructions, which takes long time to execute. Here, a single set of instruction is covered in multiple steps; each instruction set has more than three hundred separate instructions. Most instructions are completed in two to ten machine cycles. In CISC, instruction pipelining is not easily implemented forming multi-step operations.

**COMPARISON BETWEEN RISC AND CISC IS GIVEN BELOW:**

<table>
<thead>
<tr>
<th></th>
<th>RISC</th>
<th>CISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronym</td>
<td>It stands for ‘Reduced Instruction Set Computer’.</td>
<td>It stands for ‘Complex Instruction Set Computer’</td>
</tr>
<tr>
<td>Definition</td>
<td>The RISC</td>
<td>The CISC</td>
</tr>
<tr>
<td></td>
<td>processors have a smaller set of instructions with few addressing nodes.</td>
<td>processors have a larger set of instructions with many addressing nodes.</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Memory unit</td>
<td>It has no memory unit and uses a separate hardware to implement instructions.</td>
<td>It has a memory unit to implement complex instructions</td>
</tr>
<tr>
<td>Program</td>
<td>It has a hard-wired unit of programming.</td>
<td>It has a micro-programming unit.</td>
</tr>
<tr>
<td>Design</td>
<td>It is a complex compiler design.</td>
<td>It is an easy compiler design.</td>
</tr>
<tr>
<td>Calculations</td>
<td>The calculations are faster and precise.</td>
<td>The calculations are slow and precise.</td>
</tr>
<tr>
<td>Time</td>
<td>Execution time is very less.</td>
<td>Execution time is very high</td>
</tr>
<tr>
<td>External memory</td>
<td>It does not require external memory for calculations.</td>
<td>It requires external memory for calculations.</td>
</tr>
<tr>
<td>Pipelining</td>
<td>Pipelining does function correctly.</td>
<td>Pipelining does not function correctly.</td>
</tr>
<tr>
<td>Stalling</td>
<td>Stalling is mostly reduced in processors.</td>
<td>The processors often stall</td>
</tr>
<tr>
<td>Code expansion</td>
<td>Code expansion can be a problem.</td>
<td>Code expansion is not a problem.</td>
</tr>
<tr>
<td>Disc space</td>
<td>The space is saved.</td>
<td>The space is wasted.</td>
</tr>
<tr>
<td>Applications</td>
<td>Used in high end applications such as video processing, telecommunications and image</td>
<td>Used in low end applications such as security</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>processing.</th>
<th>systems, home automations, etc.</th>
</tr>
</thead>
</table>

- This given above is the comparison between RISC and CISC tabular form i.e., comparison between two on the basis of memory unit, program, design, calculations, time, external memory, pipelining, stalling, code extension and disc s
Consider the the program fragments:
- CISC: \texttt{mov ax, 10}
  \texttt{mov bx, 5}
  \texttt{mul bx, ax}
- RISC: \texttt{mov ax, 0}
  \texttt{mov bx, 10}
  \texttt{mov cx, 5}
  \texttt{Begin add ax, bx}
  \texttt{loop Begin}

  The total clock cycles for the CISC version might be:
  \((2 \text{ movs} \times 1 \text{ cycle}) + (1 \text{ mul} \times 30 \text{ cycles}) = 32 \text{ cycles}\)

  While the clock cycles for the RISC version is:
  \((3 \text{ movs} \times 1 \text{ cycle}) + (5 \text{ adds} \times 1 \text{ cycle}) + (5 \text{ loops} \times 1 \text{ cycle}) = 13\)

CONCLUSION:
To conclude from the above comparison of risc and cisc (reduced instructions set computer and complex instructions set computer resp.) we get that risc is more preferable in terms of complexity that is from the above example we get that same instructions executes using risc takes less no. of cycles as compared to cisc other comparison of both describe above

References:
1. www.wikipedia.com