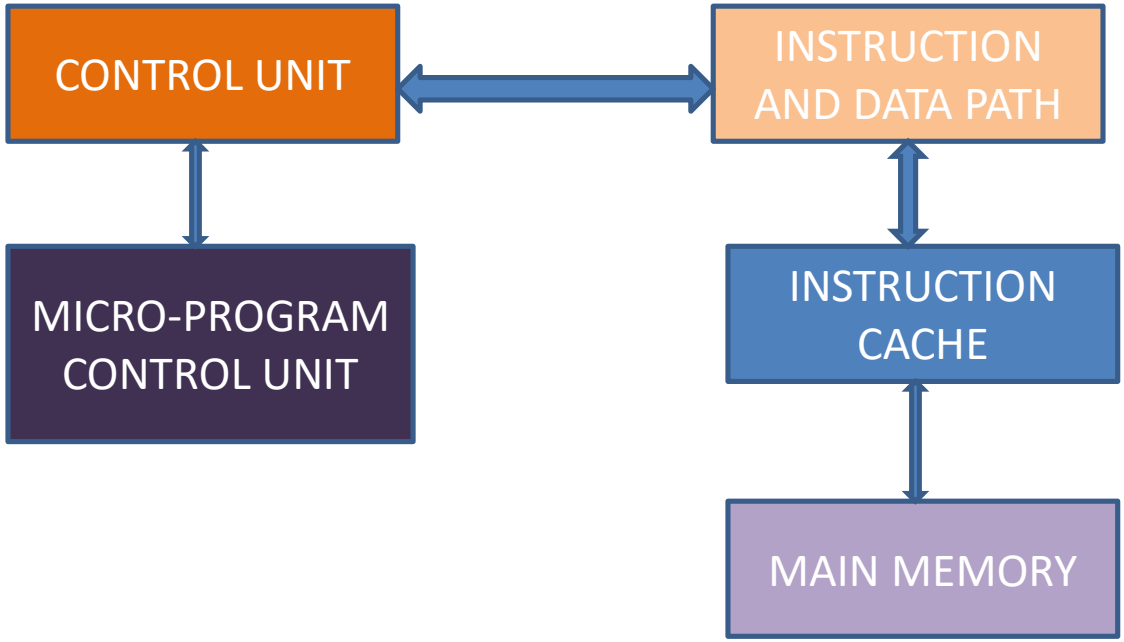
Topic 1.2 CISC and RISC processors : CISC PROCESSORS

COMPLEX INSTRUCTION SET COMPUTER



BASIC BLOCK DIGRAM OF CISC PROCESSOR

CISC : Complex Instruction Set Computer

- In early 70 , due to limitations of the technology, the design of the processor architecture was restricted due to LIMITATIONS on size of the memory.
- All the operation to be performed by the computer was to be carried out with the help of the available memory.
- So the processors were designed which could perform the COMPLEX operations. So the instruction set became COMPLEX .
- Due to the limitations in the technology, DESIGN and USE of the memory was costly. So instead of High level language/and compilers , assemblers Assembly language was preferred. Which needed less memory space.
- This led to design the more complex instructions in place of the High Level Language!!!
- But later on It was found that, because of the use of the Complex Instruction the speed of the processor drastically reduced !!!!
- As per the CISC Architecture, there is common memory for data and instruction , it goes to the control unit and then the micro program control unit generates the related code.
- For each instruction processor has to execute on Op-code (instruction part) and Operand(Data Part) from same memory!!

With the increase in the technology more powerful instructions were required to be developed for various operations for

- •Floating point arithmetic instructions.
- •Instructions for various addressing modes…
- •Assembly language instructions which are more closer to the high level instructions.

For these requirements CISC processor was developed.

Benefits

- ❑Easier programming in assembly language
- ❑Easy compilation for high level language
- ❑Less instruction for complex programming (MUL, DIV)

Question on CISC arised on the following points ……

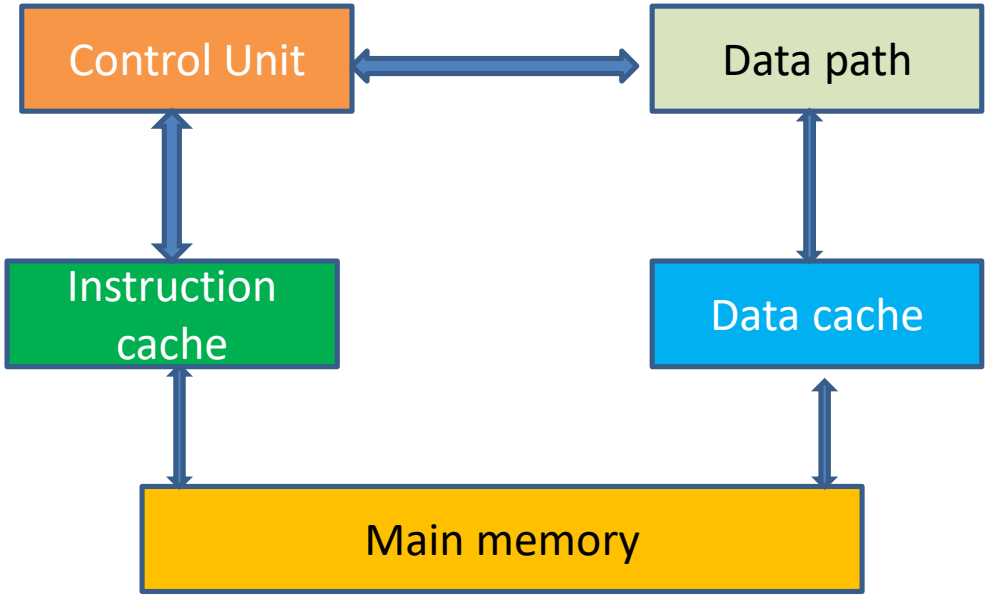
- ❖But do the programmer use all these complex instructions?
- ❖Whether all the time complex instructions are needed ??
- ❖Whether the processor which was designed to suit for these complex instructions really utilized for these complex operations???
- ❖Is this is justified for the speed of operation of a computer ???? Give your answer..

To sacrifice for the speed of operation of the computer and under utization of the CPU hardware was a question.

To overview on this : Following remedies were suggested

- To reduce the number of instruction, and there by make the CPU SIMPLER
- Get rid of the complex instructions , use simple instruction
- Use simple addressing modes so that less time will be required for accessing the operand part of the instructions of the path of this may be different !!
- Limit the number of access to the memory
- Limit the execution of instruction only for one clock cycle, else don't use that instruction
- Process more instruction, through pipe line to enhance the speed

On this basic background other Architecture was developed and it is RISC
i.e. REDUCED INSTRUCTION SET COMPUTER



Basic block diagram of RISC architecture

REDUCED INSTRUCTION SET COMPUTER : MAIN FEATURES

-Limited number of instructions in the instruction set (40-45 as compared with CISC 80-100)

-Every instruction is meant for one operation i.e no complexity

-Instructions are executed in one clock period only (exception of LOAD / STORE).

-Interaction with the memory is less. (Exception LOAD AND STORE)

Load: Reading Data from memory to register

Store : Storing data from register to memory

These two instruction requires two access for read code/read-write data

-Instructions have fixed format : Fixed length .

-Intensive use of pipeline : Each instruction have same number of pipeline stages

-Increased set of registers 32-64

- This enhances the speed of operation
- Only two instructions are memory related

-Use of multiple register set

- Fast context switching(from one set to the other set of register)
- Use of registers as windows

REDUCED INSTRUCTION SET COMPUTER : MAIN FEATURES contd..

Writing program for application in RISC BASED processor is difficult in assembly language. as

- -Instructions are not powerful.
- -Instructions have simple addressing mode
- -Instructions are specific for the specific operation.
- -Program in RISC is more optimized than that in the CISC

More programming efforts are required :

- -But this can be simplified by the use of the standard/relevant compilers.
- -Once the programming is done then the executions of the operation is faster

Processor is built with the Hardware only :

- -No microprogramming is required that is in the case of CISC , means that the instructions are decoded and executed by the built in hardware. This also enhance the speed of operation.

COMPARISON BETWEEN RISC AND CISC

Parameters	RISC	CISC
Type of instruction	Simple	Complex
Number of instructions	Less 40-50	80-100
Format of instruction	Fixed	Variable
Duration of instruction	One clock cycle	More than one clock cycle
Instruction Execution	Pipeline is possible	Pipeline is not possible .
Addressing Modes	Simple	Complex
Instructions accessing the memory	Only two Load and Store	All from the instruction set
Register set	More than one ; multiple	Only one i.e unique
Complexity	Lies in the Compilers	In micro-program CPU

Execution time of the computer:

Execution time = number of instructions x clock cycle per instruction x Tclk

For CISC : Number of instructions are less , clock cycle are more , timer clock is long

For RISC: Number of instructions are more , clock cycle one , timer clock is short

Now the question is

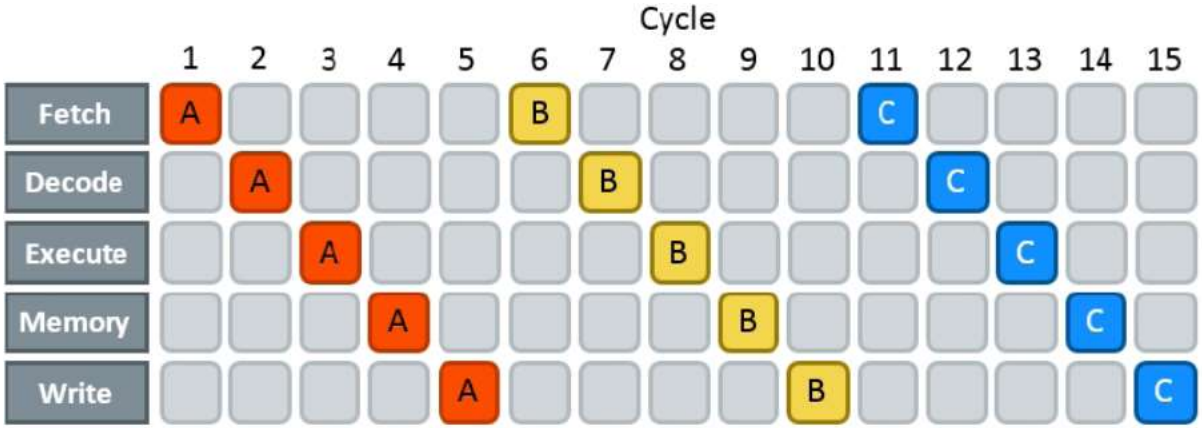
Which is better CISC OR RISC

COMBINATION OF BOTH MAY BE THE SOLUTION

Example of CISC AND RISC operation :
Multiply two numbers :and store the result in the A
In CISC: MUL AB

IN RISC : LOAD R1,ADDRESS
LOAD R2,ADDRESS
LOAD A, R1
PROD A,R2
STORE ADDRESS,A

Completion of task in less instructions using assembly is more important	**In RISC: Speed of operations in more important**
Emphasis on Hardware	Emphasis on software
Multi-clock /complex	Single clock/reduced
Memory to Memory ,Load-store for all	Register to RegisterLoad-store independent
Difficult for pipeline	Easy for pipeline



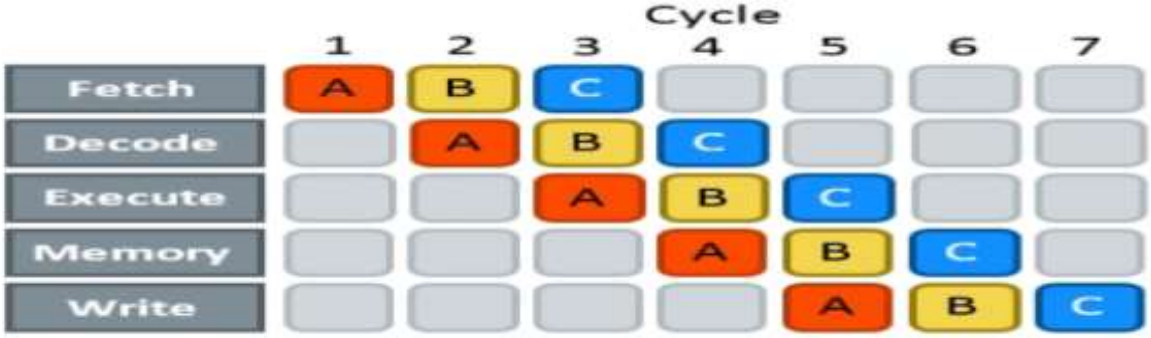


Figure 2 – Pipelined instruction processing