Ch- 09: Introduction to RISC

9) Introduction to RISC.

Q) what are different measures to correct data conflict? Explain
Q) RUC VS CISC
Q) what are the major conflicts due to instruction pipeline in RISC.
Q) what are the features that distinguish RISC processors from CISC processor? Explain
Q) Register window
Q) what are main objectives of pipelining? how does it improve speed of computation.?
Q) ~~what do you mean by arithmetic pipeline~~.

9:1) RISC Fundamentals:

- Reduced instruction set computer.
- The architecture with less instructions.
- ~~RISC architecture are also called LOAD/STORE architecture~~.
- The number of register in RISC is usually 32 or more.
- The First RISC CPU (MIPS 2000) has 32 general purpose registers
- ~~older computers/architecture~~ is CISC

~~RISC VS CISC~~

Features of RISC:

1) **Fixed length instructions** - The instructions are of same size. if there is 8 bit instruction, some instruction take these 8 bits as opcode and other might take 8 bits for opcodes or address.

2) **Limited loading and storing instructions access memory**
- RISC processors has limited interaction with memory to load & store data.

For example: if a value from memory is to be AND with accumulator.

[value1] AND [A]

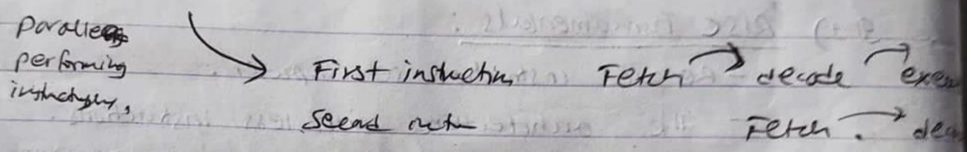
Then CPU first loads the value into a register & perform AND operation.

[B] ← [value 1]
[A] ← [A] + (B)

3) Fewer Addressing modes:

4) Instruction pipeline:

Parallel performing instructions,



5) Large number of registers.

6) H/w control unit

RISC VS CISC

RISC	CISC
1) Less number of instructions	1) Large no. of instructions
2) Relatively fewer addressing modes	2) Large variety of addressing modes.
3) Fixed length instruction format	3) Variable length instruction
4) H/w control unit	4) Microprogramed control unit
5) It requires only a single cycle for execution	5) It requires more than 1 cycle to execute instruction.
5) RISC machines contains large no. of registers	6) CISC machines doesn't have large no. of registers.

6) It has heavy pipelining	7) It has adopted pipelining but less pipelining.
7) speed of RISC machines is high.	8) Speed of CISC machine is relatively slower
8) SPARC & MIPS	9) IBM, VAX & Intel are examples.

RISC Instruction sets.

All the basic types of instructions must be represented in RISC instruction set, to perform different functions. The instructions include data movement (load, store, register move), arithmetic, shift, logic & branch instructions.

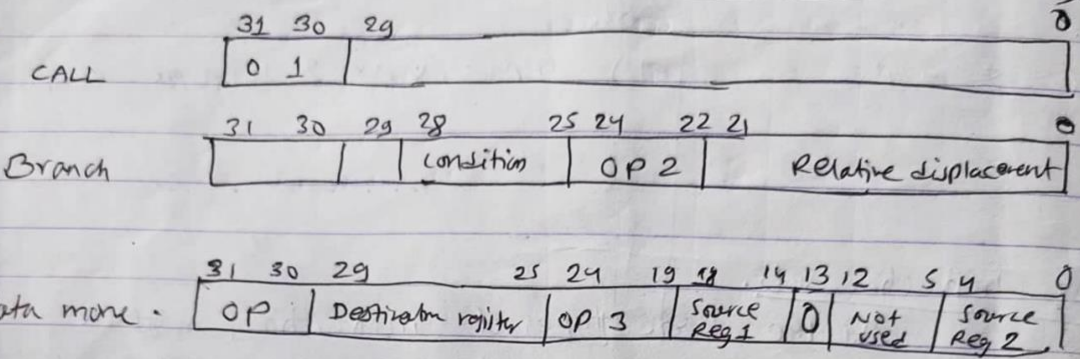
e.g: In MIPS 4000 RISC CPU, ~~there are 44 instructions~~.

Instruction type	No. of instructions
Data move	15
ALU	16
Multiply / Divide	8
Branch	25
coprocessor	11
Exception	12
special	2

RISC processors operate on two data types such as integer & float. They also operate on no. of bits within a given data type. They do not include instructions to manipulate character strings or other data type directly.

RISC processors have different instruction formats but every instruction must have same no. of bits.

→ Different instruction formats of SPARC CPU (32-bit)



Instruction Pipelining & Register window

↳ see CAD copy

Register Renaming:

Recent processor use register renaming to add flexibility to the idea of register windowing. A processor that uses register renaming can select any registers to comprise its working register "window". The CPU uses pointer to keep track of which registers are active & which physical register corresponds to each logical register.

Unlike register windowing, in which only specific groups of physical registers are active at any given time, register renaming allows any group of physical registers to be active.

Instruction pipeline conflicts / Hazards

Instruction pipeline improves overall performance but it also introduce some problems / conflicts or hazards. The conflicts arise while executing instruction in pipelining.

1) Structural Hazard / conflict. (Resource conflict)
2) Data Hazard
3) Branch Hazard

A Hazard is a potential problem that can happen in a pipelined processor. It refers to a possibility of erroneous computation when CPU tries to simultaneously execute multiple instructions which exhibit data dependence.

1) Data Hazards
   a) RAW - Read After write
   b) WAR - write After Read
   c) WAW - write after write.
2) Structural Hazards
3) Branch Hazards (control Hazards)

1) Data Hazard: Data Hazard occurs when pipeline changes the order of read/write access to operands (data).
   - due to data dependency.

EX - Execute.
MEM → store in memory
WB - write Byte to register

Consider the following execution of instructions

Instructions \ Stages	1	2	3	4	5	6	7	8	9
ADD R1, R2, R3	IF	ID	EX	MEM	WB				
SUB R4, R5, R1		IF	ID_{SUB}	EX	MEM	WB			
AND R6, R1, R7			IF	ID_{AND}	EX	MEM	WB		
OR R8, R1, R9				IF	ID_{OR}	EX	MEM	WB	
XOR R10, R1, R11					IF	ID_{XOR}	EX	MEM	WB

All the instructions after ADD use the result of ADD instruction (in R1). The ADD instruction writes the value of R1 in WB stage.

But: SUB instruction reads the value during ID stage (ID_{SUB}). This problem is data hazard.

RAW: j tries to read a source before i writes to it, so, j incorrectly gets wrong value.

WAW: j tries to write an operand before it is written by i

WAR: j tries to write a destination before it is read by i so, i incorrectly gets new value.

But RAR (read after read) is not a hazard.

2) Structural Hazard:

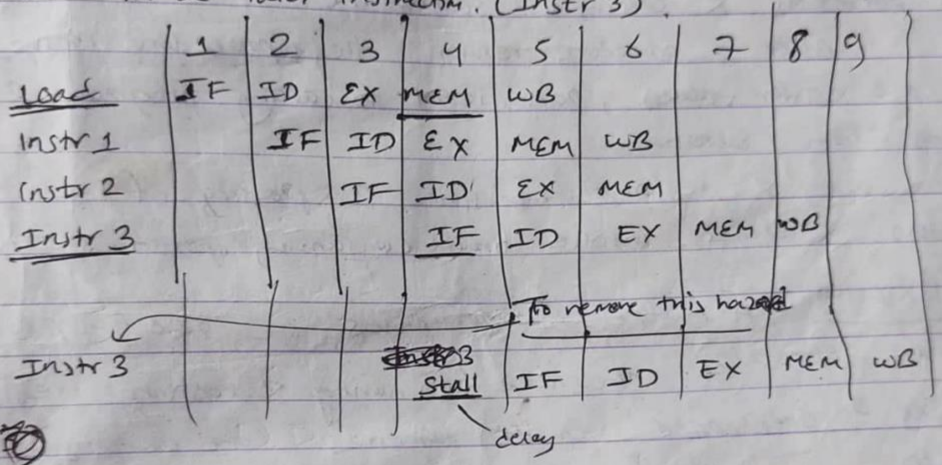
A structural hazard occurs when a part of processor's hardware is needed by two or more instructions at the same time.

If some combination of instructions cannot be accomodated because of resource conflict, the machine is said to have a structural hazard.

example 1: ~~Some~~ a machine may have only one register-file write port but in some cases the pipeline might want to perform two writes in a clock cycle.

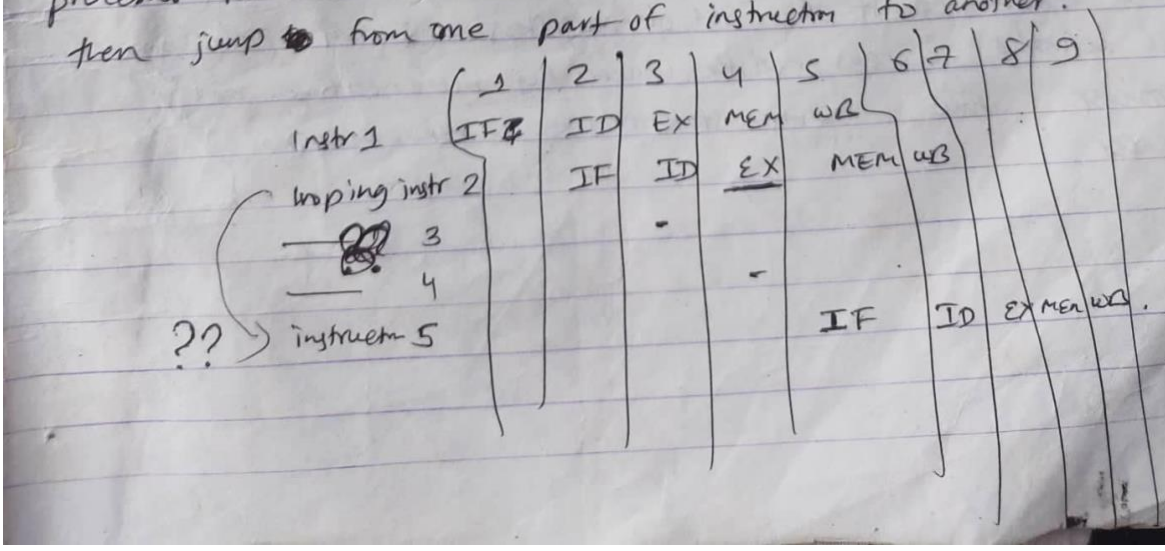
example 2:

a machine has shared a single memory pipeline for data & instructions. As a result, when an instruction contains a data memory reference (Load), it will conflict with instruction reference for a later instruction. (Instr 3).



3) Branch Hazard :-

Branch Hazard occurs when the processor is told to branch i.e if a certain condition is true, then jump from one part of instruction to another.



A group of registers banks.

Overlapped Register window : A characteristics of RISC processor is the use of overlapped register window

Problem : Procedures call & return occurs mostly in high programming language. when translated into machine language, a procedural call produces a sequence of instructions that save register to pass parameters & calling of subroutine.

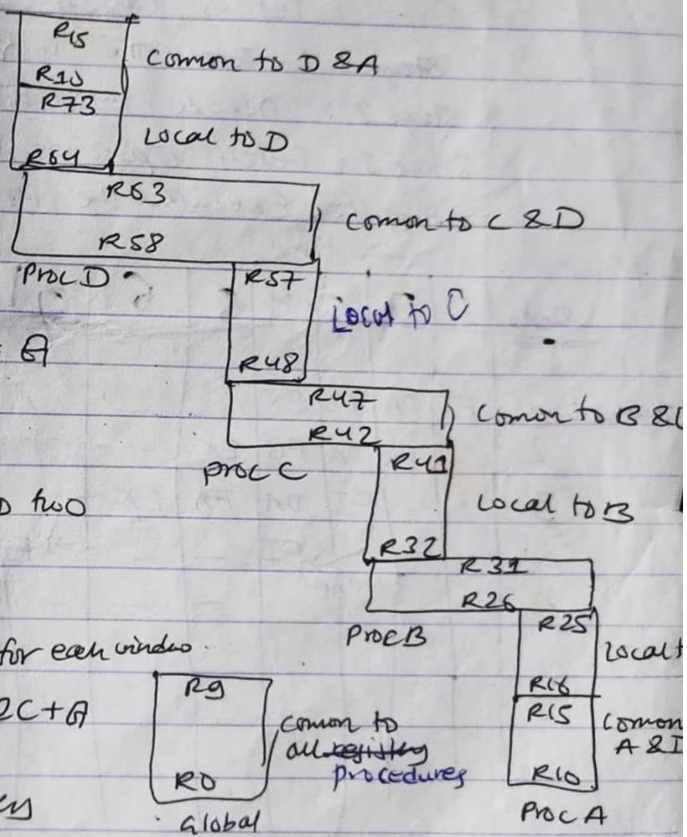
After a procedure return, the procedure restores the old register values, passes result to calling program & returns from subroutine.

saving & restoring registers & passing of parameters & results involve time consuming operation.

Solution : A multiple register banks are used by procedure which eliminates the need for saving & restoring register values - Overlapped register windows are used to provide the passing of parameters & avoids need for saving & restoring register values

Each procedure call result in allocation of new window consist of set of registers. Each procedure call activates a new register window by incrementing a pointer, while return statement decrements the pointer.

Example :- The system has a total of 74 registers. Registers R0 to R9 are Global registers – holds parameters shared by all procedures. The other 64 registers are divided into 4 windows to accomodate procedures A, B, C & D. Each register window consist of 10 local registers & two sets of 6 registers common to adjacent windows.



No. of Global registers = G

No. of local registers in each window = L

No. of common registers to two windows = C

no. of windows = W

The no. of registers available for each window.

window size (S) = L + 2C + G

The total no. of registers needed in processor is register file

$N = (L+C)W + G$

In above example $G = 10$, $L = 10$, $C = 6$ & $W = 4$

$S = 10 + 12 + 10$
$= 32$ registers.

$N = (10+6)4 + 10$
$= 74$ registers