Computer Architecture

- Report of Assignment-2

Explaination of non pipelined code

Instruction_Memory stores the instructions in the format [Address, Instruction], and Data_Memory stores data in the format [Address, Data].

→ Register file contains a list of lists containing the register number, register name, and the initial value stored in the register.

→ Below is the machine code of sorting:

```
00100000000100010000000000000000
00100000000100100000000000000000
00100001010011110000000000000000
00100001011110000000000000000000
00010010000101001000000000000110
10001101010101110000000000000000
10101101011101110000000000000000
00100001010010100000000000000100
00100001011010110000000000000100
00100010001100010000000000000001
00001000000100000000000000010111
00100001111010100000000000000000
00100011000010110000000000000000
00100000000101110000000000000000
00100000000100010000000000000000
00010010000101001000000000010000
00100010001100100000000000000001
00001100000100000000000001101010
00100001011110000000000000000000
00000011000011001100000000100000
10001111000101000000000000000000
00010010010010010000000000001000
00001100000100000000000001110000
00100001011110000000000000000000
00000011000011011100000000100000
10001111000101010000000000000000
00000010100101011011100000100010
00011110111000000000000000110000
00100010010010010000000000000001
00001000000100000000000000101000
00100010001100010000000000000001
00001000000100000000000000100010
00100000000101110000000000000000
00100010100011110000000000000000
00100010101101000000000000000000
00100001111101010000000000000000
00100001011110000000000000000000
00000011000011001100000000100000
10101111000101000000000000000000
00100001011110000000000000000000
00000011000011011100000000100000
10101111000101010000000000000000
00100010010010010000000000000001
00001000000100000000000000101000
00100000000011000000000000000000
00000011001000101100000000100000
00000011001000101100000000100000
00000011001000101100000000100000
00000011001000101100000000100000
00000011111100000000000000001000
00100000000011010000000000000000
00000011011001001101000001000000
00000011011001001101000001000000
00000011011001001101000001000000
00000011111100000000000000001000
```

→ CLOCK is initialized to zero, which will be used to count clock cycles.
→ The different functions have various uses

1. Binarytodecimal- converts a binary string to its decimal equivalent. It's used for converting binary values to decimal, particularly for the immediate (imm) field in instructions.
2. Getregvalue- the value stored in a register based on its register number.
3. Setregvalue - sets the value of a register based on its register number.
4. Getdatamemvalue - the value stored in data memory at a specified memory address.
5. Setdatamemvalue - sets the value in data memory at a specified memory address.
6. GetAddressindex - returns the index of an instruction in the Instruction_Memory list based on its memory address.
7. Executeinstruction is the main function which takes the instruction and its address as parameters, then it breaks the instruction into different parts: opcode, rs, rt, rd, shamt, imm fields according to the type of instruction.
    - After breaking the instruction it performs the execution based on the given opcode and returns the value(s) needed.
    - The code then proceeds to read a set of machine code instructions, initialize memory and register values, and execute the instructions in a loop until a specified ending address is reached.One by one instructions in the machine code are executed and deleted after the execution, so when the length of machine code is equal to 0 the while loop exits.
    - Starting address and ending address are taken input by the user and the value is given to pc where the 1st instruction is executed then accordingly, pointer gets updated after execution of each instruction.
    - The code is executed till the ending address which contains last instruction to be executed in machine code.All the changes in values of registers and data in data memory are updated in their corresponding lists.
    - While loop of flag is the main function from where execution is started. Instructions are executed till the pointer doesn't point to the ending address, and the flag becomes false to exit the loop and program ends

INPUT:

4194380

33

4194684

23

4194380

4194508

268501184

5

5 4 3 2 1

268501216

OUTPUT:

```
[[268501184, 3], [268501188, 2], [268501192, 4], [268501196, 6], [268501200, 1], [268501216, 1], [268501220, 2], [268501224, 3], [268501228, 4], [268501232, 6]]
```

Explaination of Pipelined code

Firstly, we initialize pipelined registers If_Id, Id_Ex, Ex_Mem, Mem_Wb as empty list. Then different functions perform various tasks such as

→ CalculateStalls(DependentInstructionNo, InstructionNo)

- It calculates the number of stalls required based on the difference between the dependent and current instruction numbers and updates the InstructionStalls list to show stalls for each instruction. Lastly it returns the calculated number of stalls.

→ Binarytodecimal- converts a binary string to its decimal equivalent. It's used for converting binary values to decimal, particularly for the immediate (imm) field in instructions.

→ Getregvalue- the value stored in a register based on its register number.

→ Setregvalue - sets the value of a register based on its register number.

→ Getdatamemvalue - the value stored in data memory at a specified memory address.

→ Setdatamemvalue - sets the value in data memory at a specified memory address.

→ IF: it fetches the instruction at the specified instruction pointer from Instruction_Memory and returns the instruction as a list [Address, Instruction]

→ ID: it takes the instruction and tval as parameters where tval is a boolean representing the availability of the instruction (used for stalling).

- It decodes the binary instruction and generates control signals based on the opcode.

- For R-type instructions, extracts register values and for II-type instructions it extracts register values and immediate values. Lastly it returns a list [CS, rs, rt, rd, imm, MemRead, MemWrite, MemtoReg, RegWrite]

→ EX: It takes the values returned by ID function as parameters and Performs the ALU operation based on the control signals. Then Returns a list with the ALU result and control signals.

→ MEM: It takes Memory address, Control signals., Source/destination registers., Immediate value, Value to be stored in memory and Memory and register write control signals as parameters

- It Reads from or writes to the data memory based on control signals and returns a list with the result and control signals.

→ WB: It takes the value to be written back, destination register along with Register write control signal and Writes the value back to the specified register if RegWrite is enabled.

→ Breakinstruction: Determines the opcode and extracts relevant fields based on the instruction format and returns a tuple representing the instruction type and its components.

→ Flush: it resets the pipeline registers in case of a branch or jump instruction by setting the pipeline registers (IF_ID, ID_Ex, Ex_Mem) to empty lists.

→ FindDependencies: it Searches the New_Dependencies1 list for dependencies related to the given instruction pointer and Returns a tuple with information about the dependent instruction.

→ UpdateStalls: it updates the InstructionStalls list to eliminate stalls after a branch instruction.

- Instruction_Memory stores the instructions in the format [Address, Instruction], and Data_Memory stores data in the format [Address, Data].
- RegModification dictionary is used to keep track of the index (position) of the latest instruction that modified each register which will help in finding dependencies.
- InstructionStalls = Calculates the number of stalls after a particular instruction
- Starting address and ending address are taken input by the user and the value is given to pc

  where the 1st instruction is executed then accordingly, pointer gets updated after execution of each instruction.

- The code is executed till the pc becomes ending address i.e the last instruction to be executed in machine code is fetched + 8. This is because after fetching the last instruction it will take 5 clock cycles to execute and there can be at max 3 stalls.
- We can consider one loop to be one clock cycle. In that clock cycle all phases are performed once.
- All the phases return an empty list when no values are passed to it or when they are not executed.
- First the IF function fetches the 32 bit instruction.Then we check if any stalls are there or not. If there are then they are updated accordingly.
- The ID function decodes the instruction which was fetched previously and in a similar way the EX and MEM and WB functions perform.
- If the values of stalls are non-zero then IF_TVal and ID_TVal are zero.
- As a result of which the Fetch keeps fetching the same instruction and decode does not perform any tasks.
- At the end of all the phases the result of the phases are updated in the corressponding pipeline registers.
- If there are beq instruction then it checks the condition if it has to jump jumps in the Mem phase

INPUT:

4194380

2

4194380

4194396

268501184

1

5

268501216

OUTPUT:

```
[[268501184, 6], [268501216, 720]]
```