MIPS32 Pipelined Processor - Verilog Implementation

Ref: Hardware modelling using Verilog [NPTEL Prof. Indranil Sen Gupta IITKGP]

Introduction:

MIPS (Microprocessor without Interlocked Pipelined Stages**)**

It is a family of RISC (reduced Instruction Set) ISA (Instruction Set Architectures) developed by MIPS Computer Systems, now MIPS Technologies, based in the United States.

Instrsuction types:

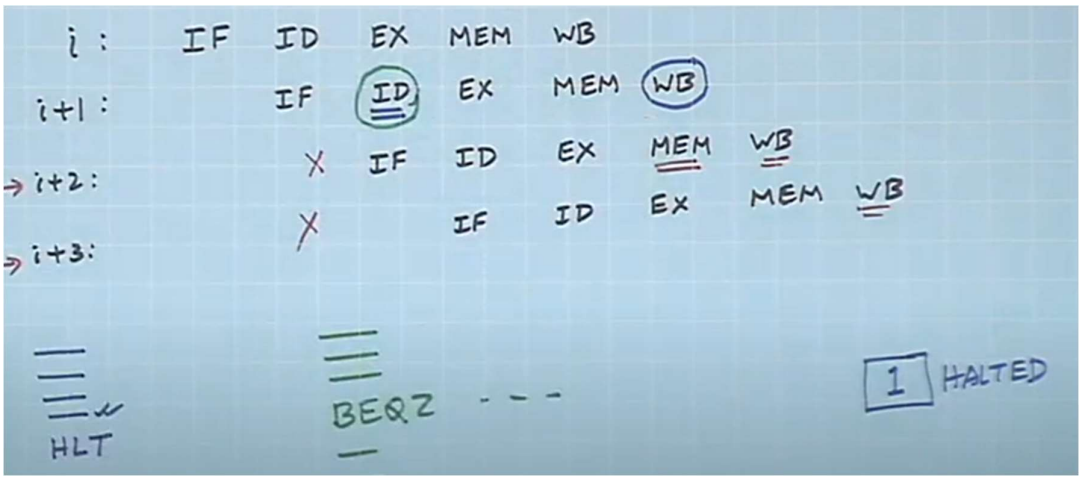
type	**-31-**		**format (bits) (31-0)**			**-0-**
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		
J	opcode (6)	address (26)				

Basic process of execution of instructions in 5 pipelined stages, which are given below:

Fetch > Decode > Execute>Memory>Write Back

Halted and Taken branch are additional 1bit variables to pipelined implementation, compared to previous.

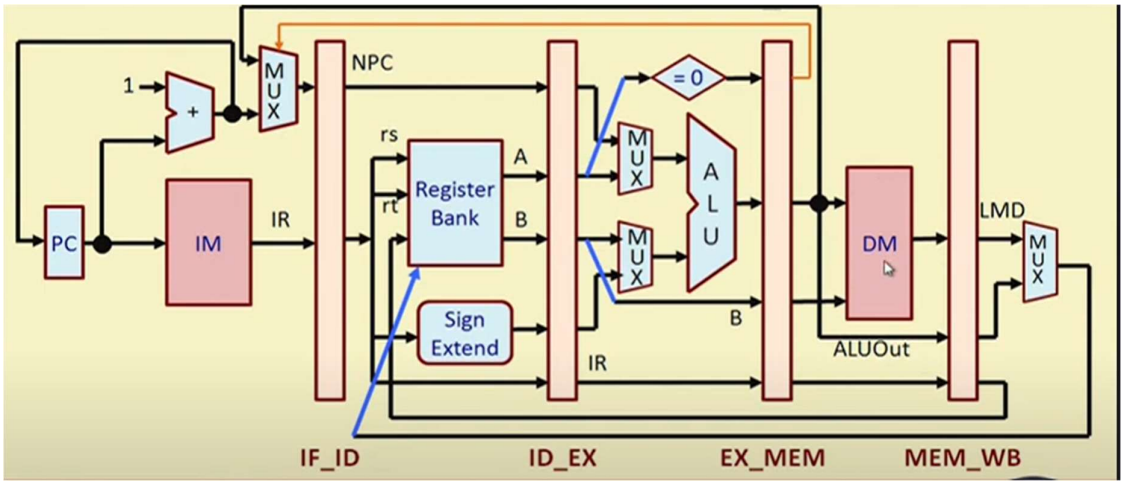
When halt is detected, we can't stop the operations right away as the previous instructions are in different other stages of pipeline. We can stop only after the previous instructions in pipe are cleared and don't write after the halt is detected. So we keep a HALTED reg for this case. Similarly for the branch taken.



Tools Used:

iverilog ,GTKwave

Architecture used for implementation:



Instructions available in this implementation:

SNO	Instructions	Code	Description
1	ADD	000000	Addition
2	SUB	000001	Subtraction
3	AND	000010	Logical and
4	OR	000011	Logical or
5	SLT	000100	Set less than
6	MUL	000101	multiply
7	HLT	111111	Halt
8	LW	001000	Load
9	SW	001001	Store
10	ADDI	001010	Add immediate
11	SUBI	001011	Subtract immediate
12	SLTI	001100	Set less than immediate
13	BNEQZ	001101	Branch not equal to zero
14	BEQZ	001110	Branch equal zero

We write different test benches for different programs.

We simply write program in memory and load from the specific location in TB.

Example 1:

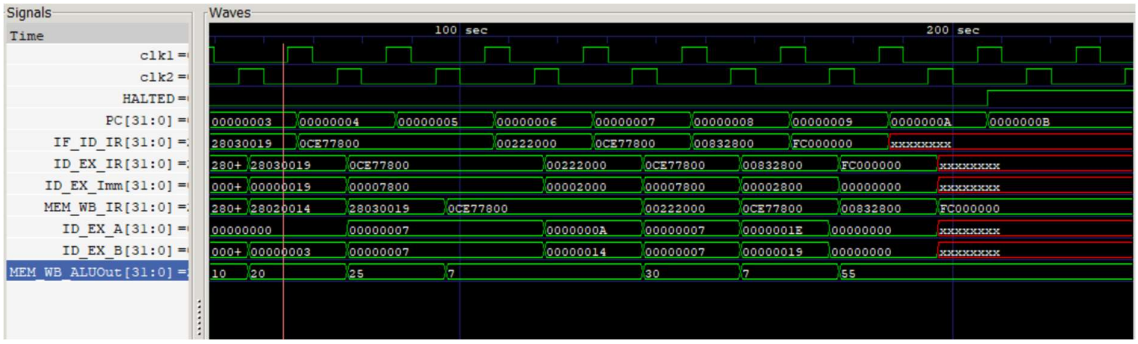
Executing the **addition** program:

Assembly Language Program	Machine Code (in Binary)
ADDI R1,R0,10	001010 00000 00001 0000000000001010
ADDI R2,R0,20	001010 00000 00010 0000000000010100
ADDI R3,R0,25	001010 00000 00011 0000000000011001
ADD R4,R1,R2	000000 00001 00010 00100 00000 000000
ADD R5,R4,R3	000000 00100 00011 00101 00000 000000
HLT	111111 00000 00000 00000 00000 000000

Test Bench Output:

```
E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>iverilog -o test tb_add.v

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>vvp test
VCD info: dumpfile mips.vcd opened for output.
R0 =  0
R1 = 10
R2 = 20
R3 = 25
R4 = 30
R5 = 55
tb_add.v:47: $finish called at 300 (1s)

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>gtkwave mips.vcd

GTKWave Analyzer v3.3.100 (w)1999-2019 BSI

[0] start time.
[300] end time.
```

Wave forms from GTK wave:



Example 2

- Load a word stored in memory location 120, add 45 to it, and store the result in memory location 121.
- The steps:
  - Initialize register R1 with the memory address 120.
  - Load the contents of memory location 120 into register R2.
  - Add 45 to register R2.
  - Store the result in memory location 121.

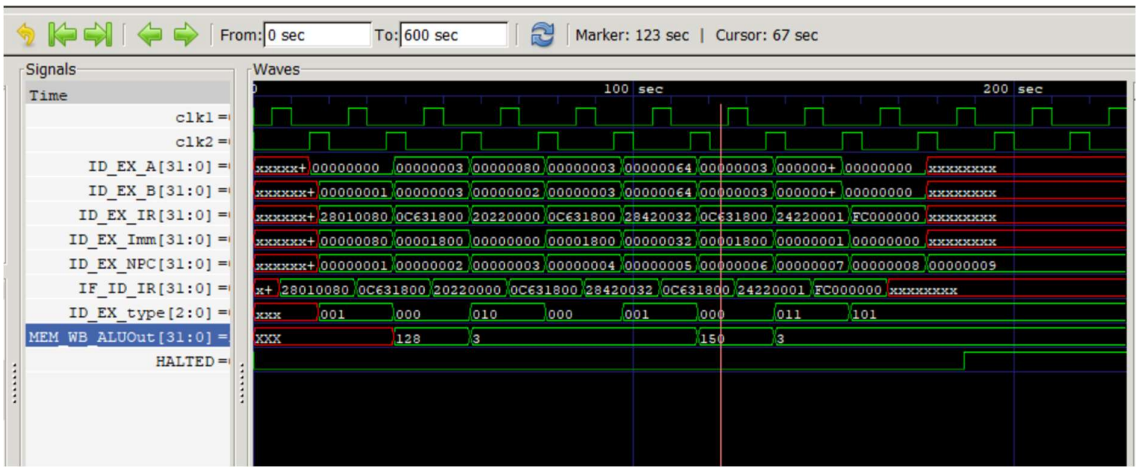
Assembly Language Program		Machine Code (in Binary)
ADDI	R1,R0,120	001010 00000 00001 0000000001111000
LW	R2,0(R1)	001000 00001 00010 0000000000000000
ADDI	R2,R2,45	001010 00010 00010 0000000000101101
SW	R2,1(R1)	001001 00010 00001 0000000000000001
HLT		111111 00000 00000 00000 00000 000000

```
E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>iverilog -o eg2 tb_ld.v

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>vvp eg2
VCD info: dumpfile mips .vcd opened for output.
Mem[120]:   85
Mem[121]:  130
tb_ld.v:42: $finish called at 600 (1s)

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>
```

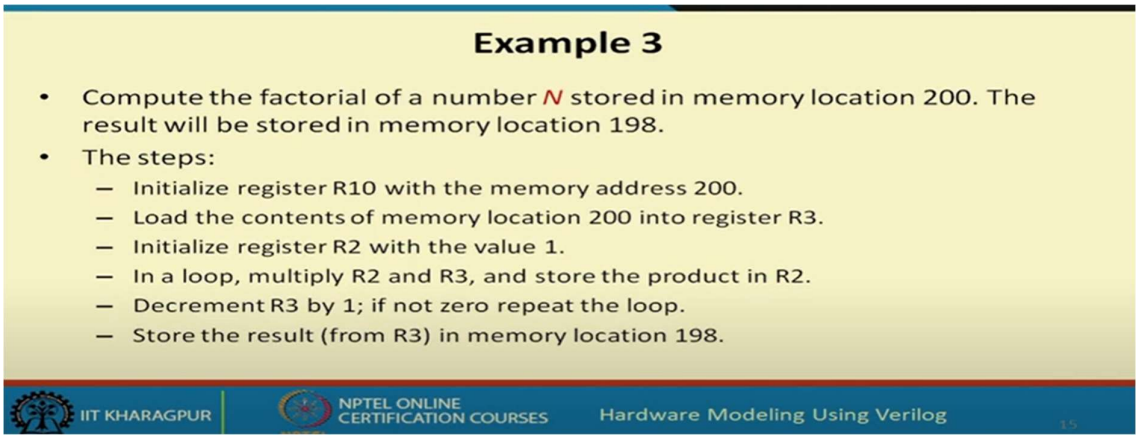
Eg2_1: Read data at 128 location and add 50 to it, and store in 129 location.

100 is stored at 128 location, adds 50 to it and stores in 129 location, which is 150.

```
E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>iverilog -o eg2_1 tb_ld.v

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>vvp eg2_1
VCD info: dumpfile mips .vcd opened for output.
Mem[128]:  100
Mem[129]:  150
tb_ld.v:48: $finish called at 600 (1s)

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>
```



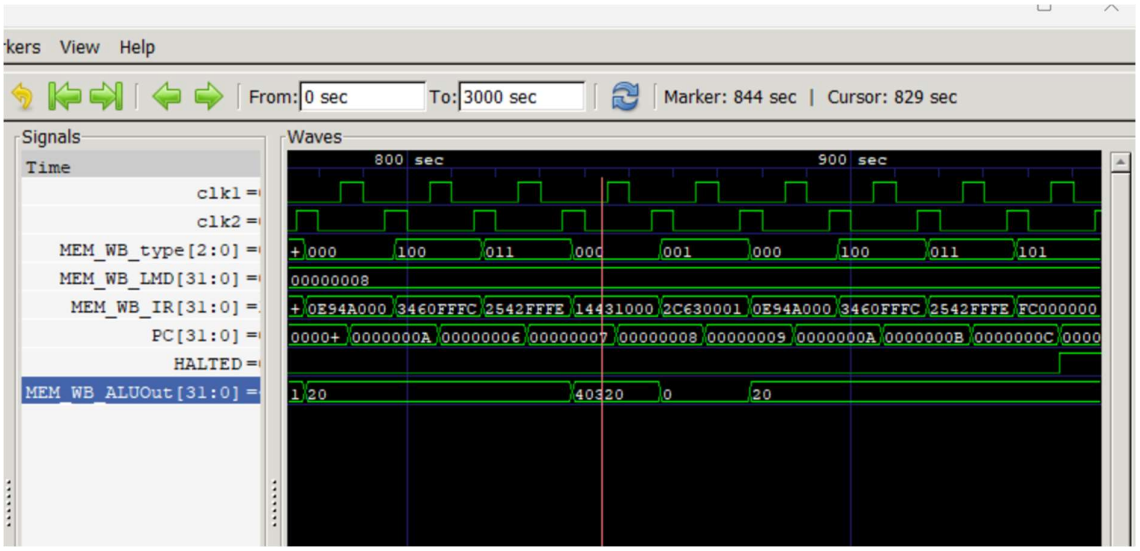
Example 3:



Assembly Language Program			Machine Code (in Binary)
	ADDI	R10,R0,200	001010 00000 01010 0000000011001000
	ADDI	R2,R0,1	001010 00000 00010 0000000000000001
	LW	R3,0(R10)	001000 01010 00011 0000000000000000
Loop:	MUL	R2,R2,R3	000101 00010 00011 00010 00000 000000
	SUBI	R3,R3,1	001011 00011 00011 0000000000000001
	BNEQZ	R3,Loop	001101 00011 00000 1111111111111101
	SW	R2,-2(R10)	001001 00011 01010 1111111111111110
	HLT		111111 00000 00000 00000 00000 000000

Finding Factorial of 8.

```
E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>iverilog -o eg3_out eg3_tb.v

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>vvp eg3_out
VCD info: dumpfile mips.vcd opened for output.
R2:    2
R2:    1
R2:    8
R2:   56
R2:  336
R2: 1680
R2: 6720
R2: 20160
R2: 40320
R2: 40320
Mem[200] =  8, Mem[198] =  40320
eg3_tb.v:44: $finish called at 3000 (1s)

E:\GOURAV\MTech MVLSI\Placements\HDL_Scripting\Verilog\Proj_verilog\MIPS32_pipelined\NPTEL>
```



Conclusion:

A very basic MIPS processor is designed in Verilog and few examples are executed with tb.

Dummy lines are used to prevent the processor from crash, due to hazards.

We can design a more robust chip which can handle hazards, more instructions can be included in future work.

References

1. MIPS architecture - Wikipedia
2. MIPS32 Architecture – MIPS (archive.org)
3. NPTEL :: Computer Science and Engineering - NOC:Hardware modeling using verilog