ADVANCED COMPUTER ARCHITECTURE ASSIGNMENT 2

RAHUL PARMANI 2019H1030021G
D MOHIT VARSHA 2019H1030026G
NSV RAM PRATHAP 2019H1030559G

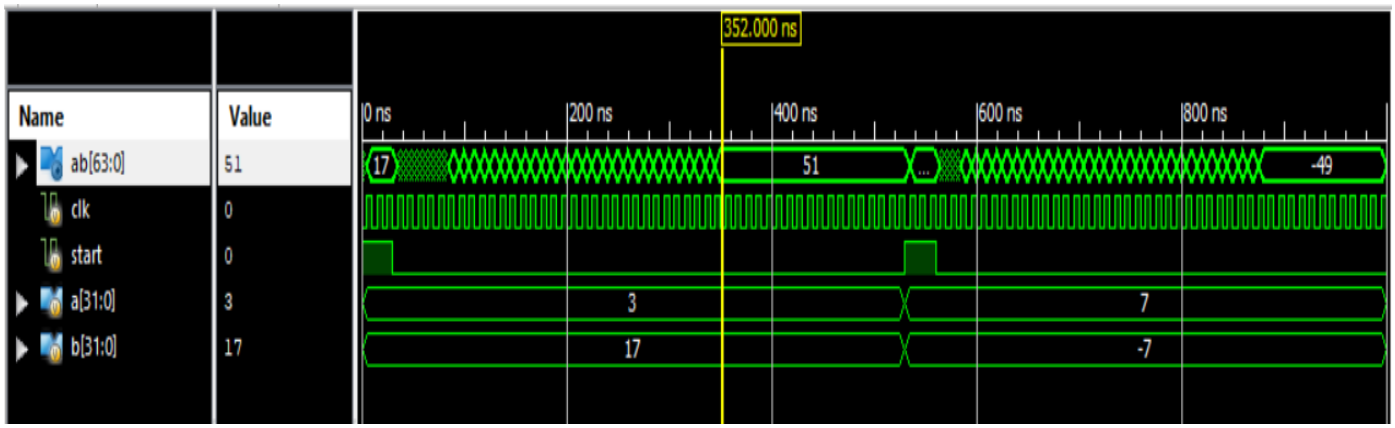
Q1. <u>32-bit Booth Multiplier</u>

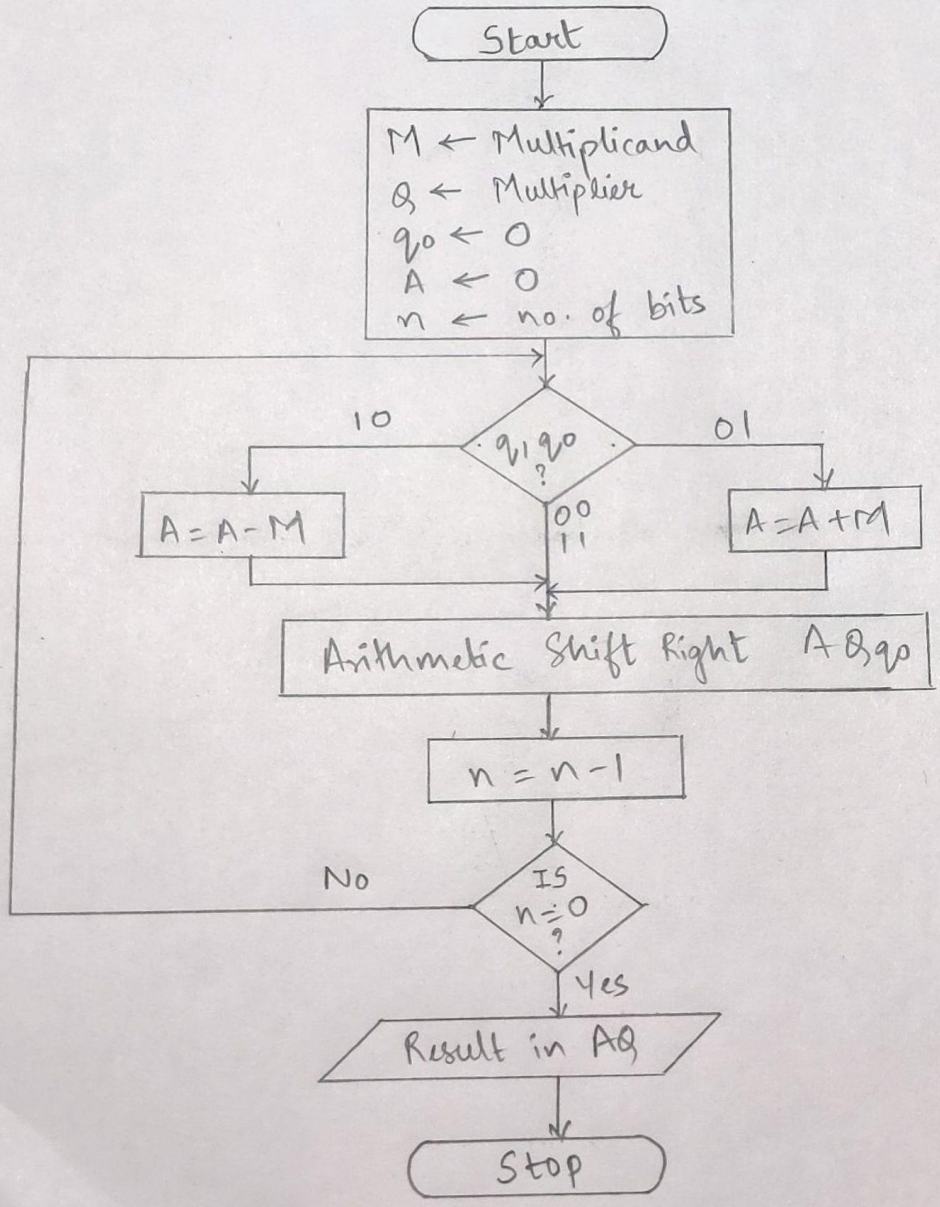
<u>Working:</u>

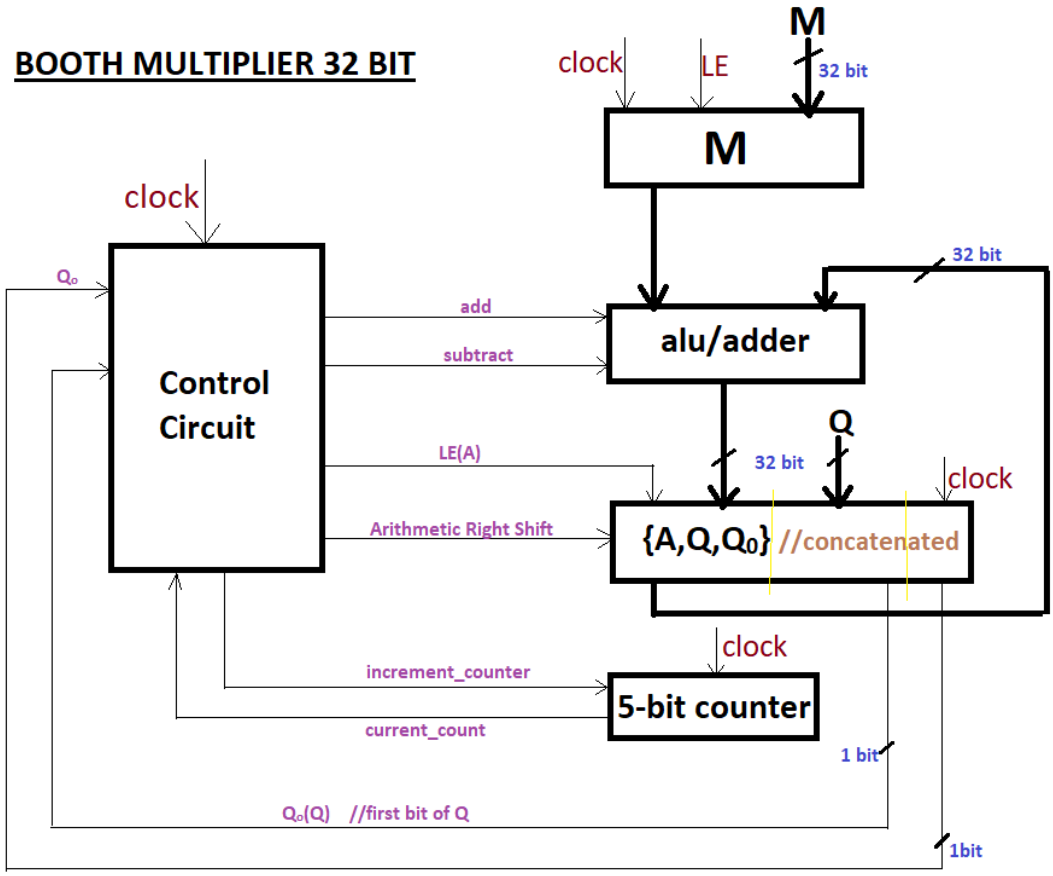
Circuit takes 4 inputs (**multiplier, multiplicand, clock, start**)
Circuit gives 1 output (**result_product**)

Variable **start** when set to 1 implies that all the necessary temporary registers will be initialized. Assuming 30ns time is taken for initialization. Then **start** variable is set to 0. Now the actual computation starts. Since the inputs are 32 bit numbers, booth multiplication takes 32 iterations to compute the result. Assuming a clock period of 10ns we need at least 320ns to compute the result. After total time of (30ns for initialization + 320ns of computation) around 350ns the final result will be out and will be stable. The final result is shown by the variable **ab,** also upon zooming in partial result at each iteration can also be observed. Following the result the second test case is given.

Since Xilinx has default simulation limit of 1000ns, a limited number of test cases are given, however other test cases case be checked by simply uncommenting and re commenting the cases in test bench. Code was tested with a combination of signed numbers both positive and negative. While verifying the outputs kindly change the radix to Signed Decimal




Start
M ← Multiplicand
Q ← Multiplier
q0 ← 0
A ← 0
n ← no. of bits
q1 q0 ?
10
01
00
11
A = A - M
A = A + M
Arithmetic Shift Right A Q q0
n = n - 1
Is n = 0 ?
No
Yes
Result in AQ
Stop



STATE TABLE

Concatenated value of {1st bit of Q , Q_0}	Operation to be performed
00	Arithmetic Right Shift of {A,1st bit of Q, Q_0}
01	A=A+M
10	A=A-M
11	Arithmetic Right Shift of {A,1st bit of Q, Q_0}

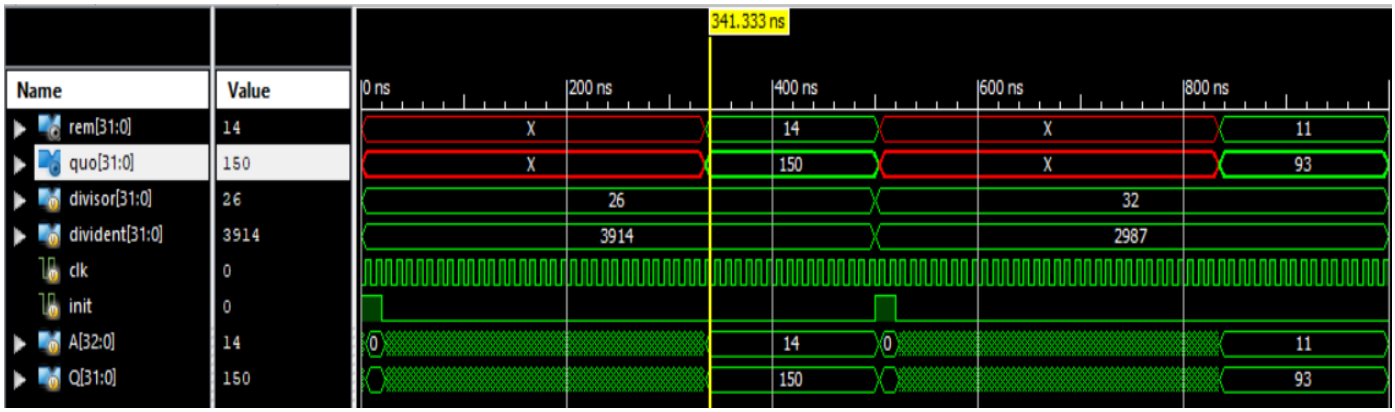
Q2. <u>32-bit Non Restoring Divider</u>

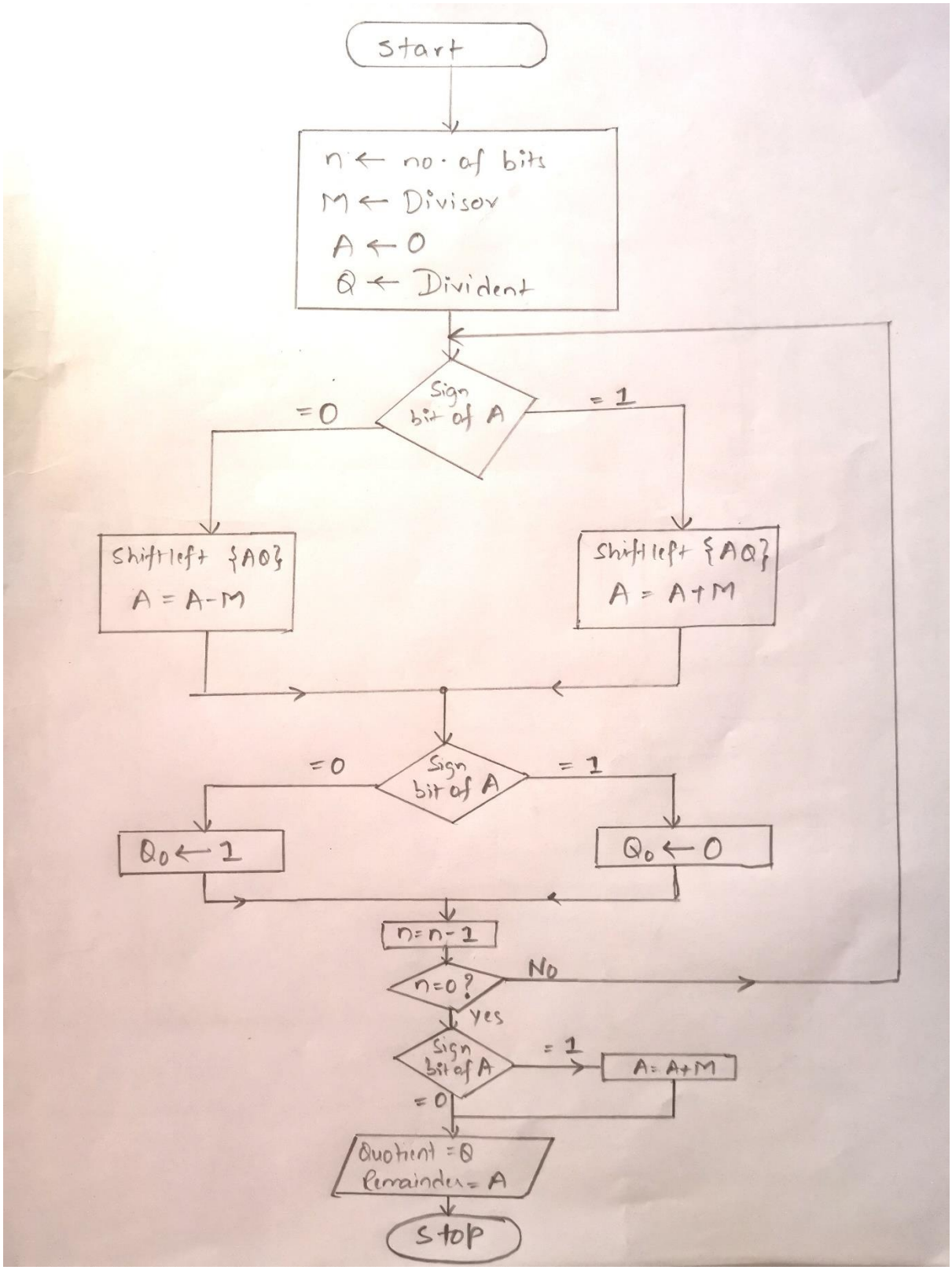
<u>Working:</u>

Circuit takes 4 inputs (**divisor, divident, clock, init**)
Circuit gives 2 outputs (**remainder, quotient**)

In the test bench, the test cases have one of the Variable **init** which when set to 1 implies that all the necessary temporary registers will be initialized. Assuming 20ns time is taken for initialization. Then **init** variable is set to 0. Now the actual computation starts. Since the inputs are 32 bit numbers, non-restoring divider takes 32 iterations to compute the result. Assuming a clock period of 10ns we need at least 320ns to compute the result. After total time of (20ns for initialization + 320ns of computation) around 340ns the final results (quotient and remainder) will be out and will be stable. Following this second test case can be given.

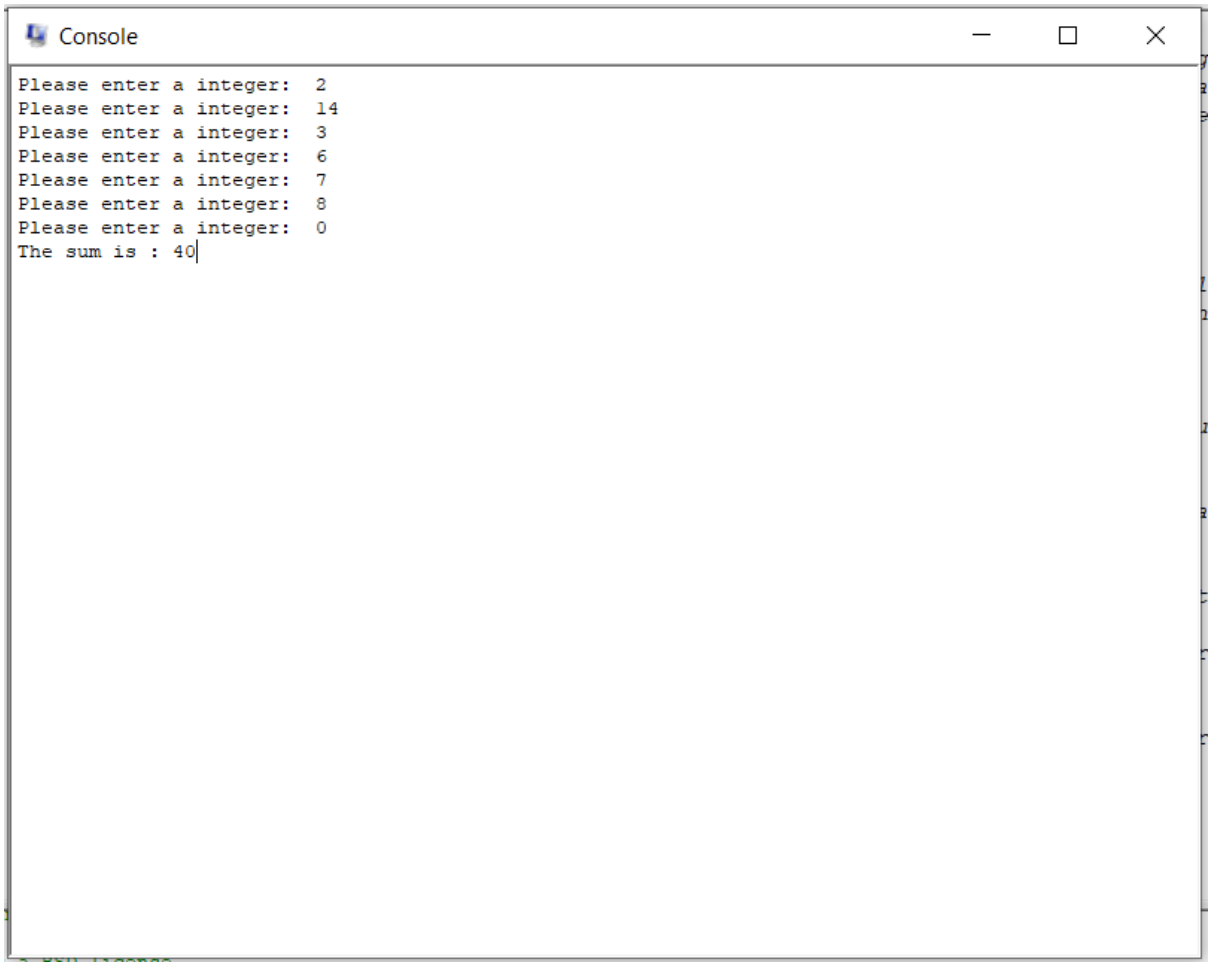
However the output **(rem** and **quo)** only display the final result only when available, until then they stay as **X.** In order to observe the intermediate results of the computation kindly drag the registers **A** and **Q** under **uut** of the simulation window. Here the assumption is unsigned integers, therefore no need make any changes to radix, assuming default radix is unsigned decimal.




Start
n ← no. of bits
M ← Divisor
A ← 0
Q ← Divident
Sign bit of A
=0
=1
Shift left {AQ}
A = A-M
Shift left {AQ}
A = A+M
Sign bit of A
=0
=1
Q0 ← 1
Q0 ← 0
n=n-1
n=0?
No
yes
Sign bit of A
=1
A= A+M
=0
Quotient = Q
Remainder = A
Stop

Q3. QTSpim program for Reading Integers and printing sum

Output:

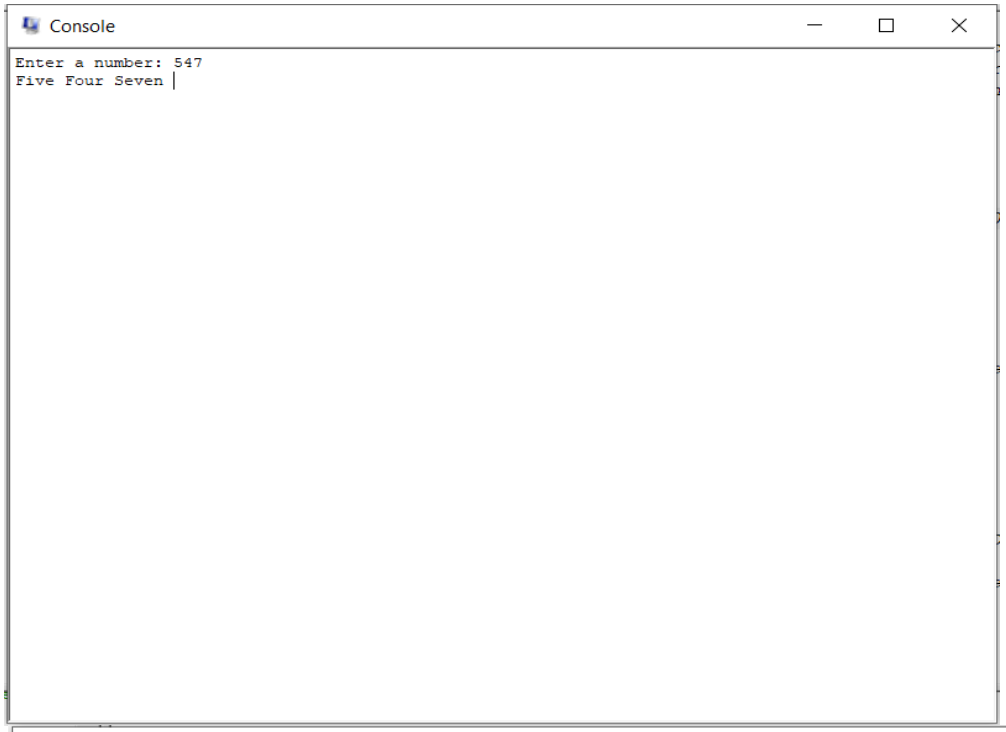


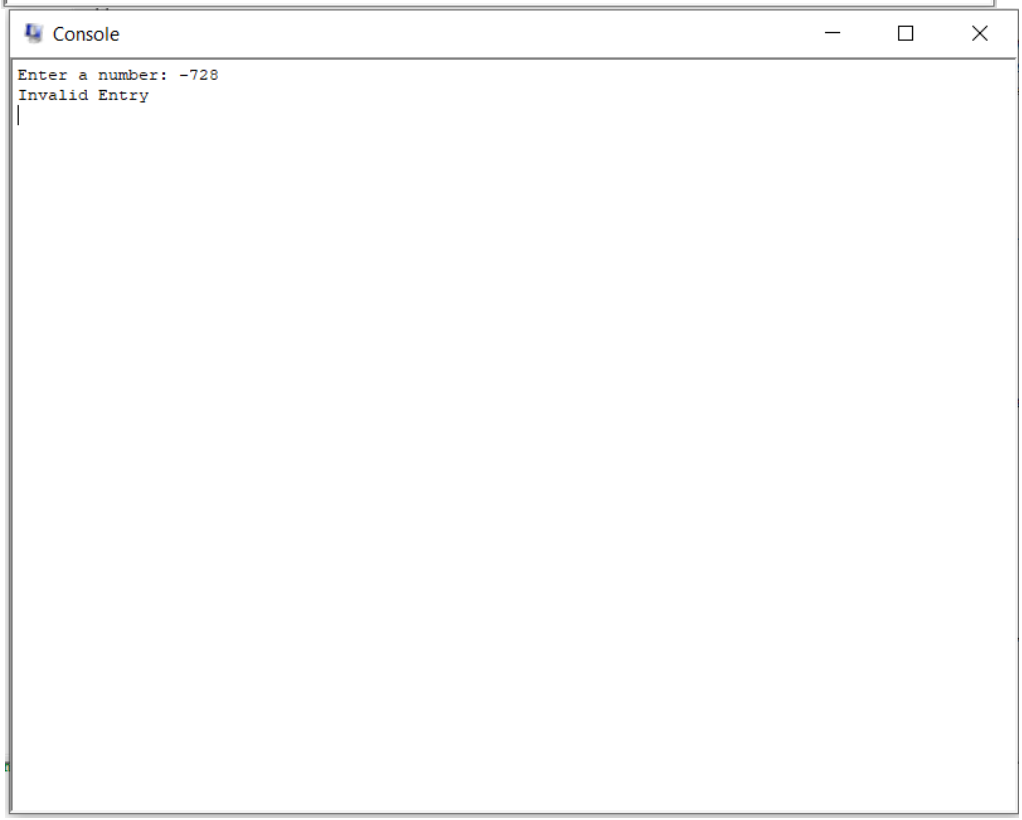
Q4. QTSpim program to print sum of largest of 2 integers for given 3 integers

Output:

Q5. QTSpim program to print names of digits of a given number

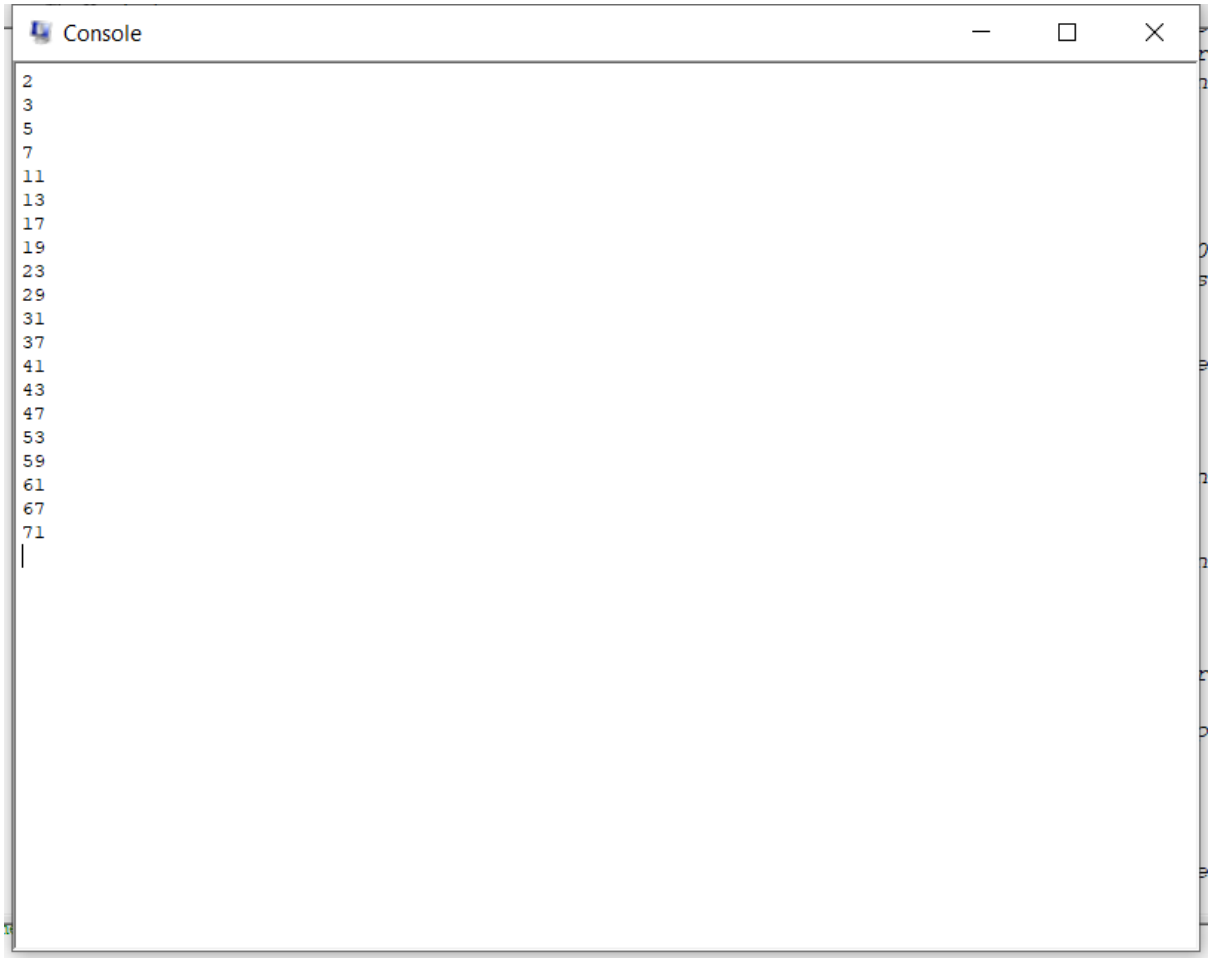
Output:





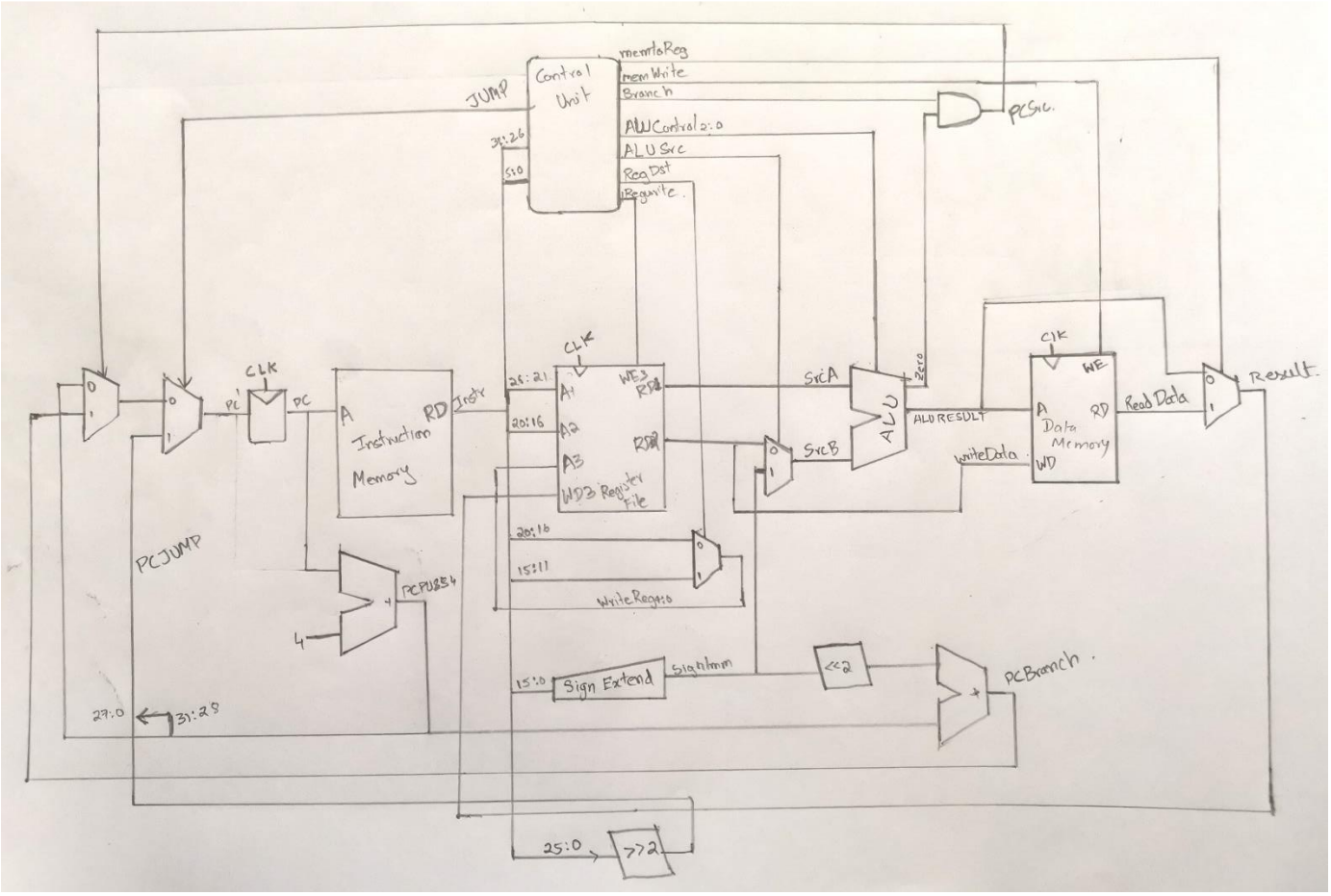
Q6. MIPS assembly language program to compute and print the first 20 prime numbers

Output:



Q7. MIPS Single Cycle Processor

Working:

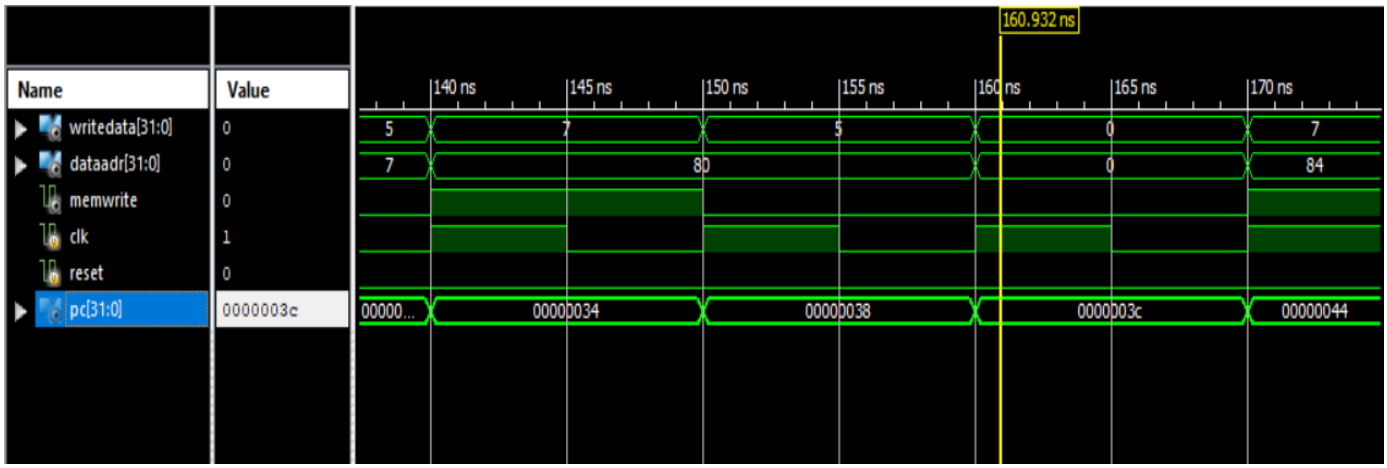


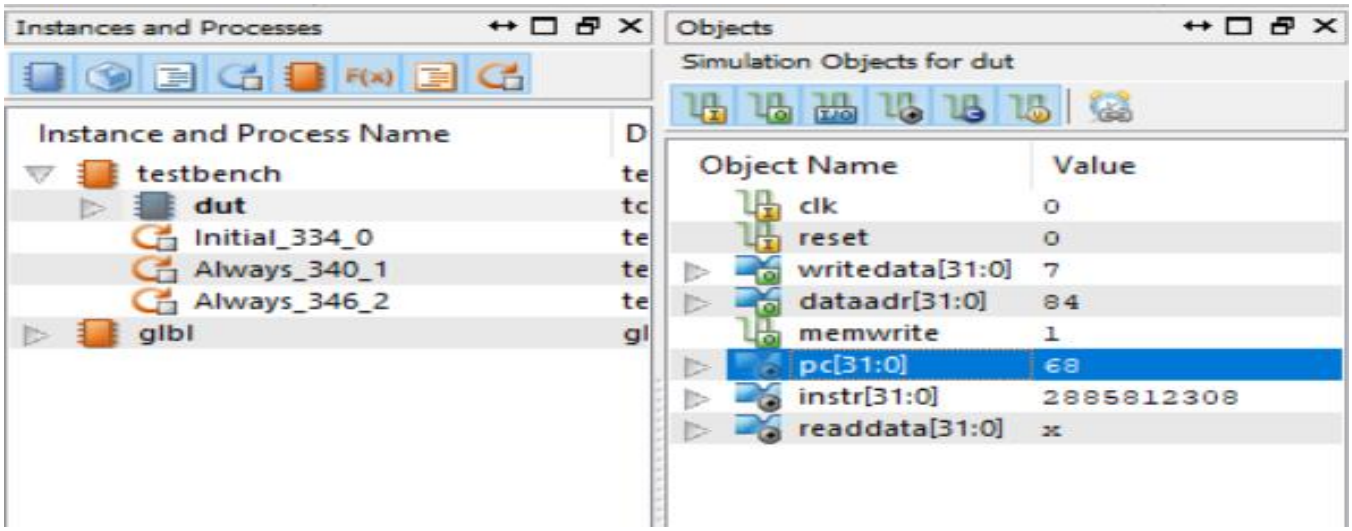
Important Note:

Once the project is created with the help of **singlecycle.v** file it is necessary to have the **memfile.dat** along with the singlecycle.v file in the **same directory** as the project. Otherwise this may lead to errors or failure in execution.

Also for verifying the **addresses** it is necessary to add the program counter **PC** to the simulation by dragging it from the **dut** part of the simulation window. It is important to observe that the values for program counter need to be in **Hexadecimal**, hence kindly change PC radix to hexadecimal and all other radix are signed decimal. Other this may lead to confusing outputs.

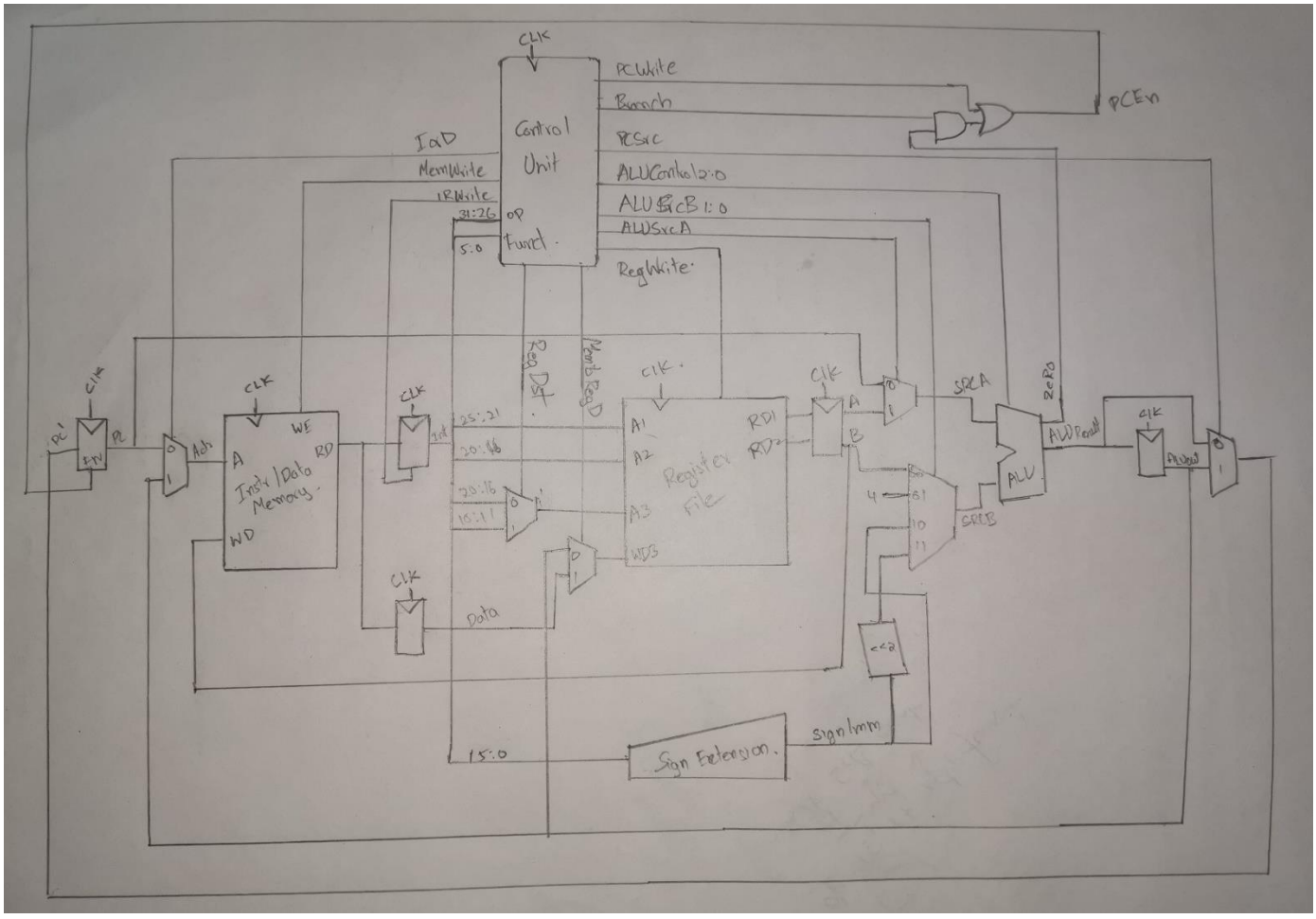
Simulation screen shots only include the last 150 to 200 ns as space is insufficient for whole simulation to be captured at one go.


160.932 ns
Name
Value
140 ns
145 ns
150 ns
155 ns
160 ns
165 ns
170 ns
writedata[31:0]
0
5
7
5
0
7
dataadr[31:0]
0
7
80
0
84
memwrite
0
clk
1
reset
0
pc[31:0]
0000003c
00000...
00000034
00000038
0000003c
00000044


Instances and Processes
Instance and Process Name
testbench
dut
Initial_334_0
Always_340_1
Always_346_2
glbl
Objects
Simulation Objects for dut
Object Name
Value
clk
0
reset
0
writedata[31:0]
7
dataadr[31:0]
84
memwrite
1
pc[31:0]
68
instr[31:0]
2885812308
readdata[31:0]
x

Q8. MIPS Multi Cycle Processor

Working:

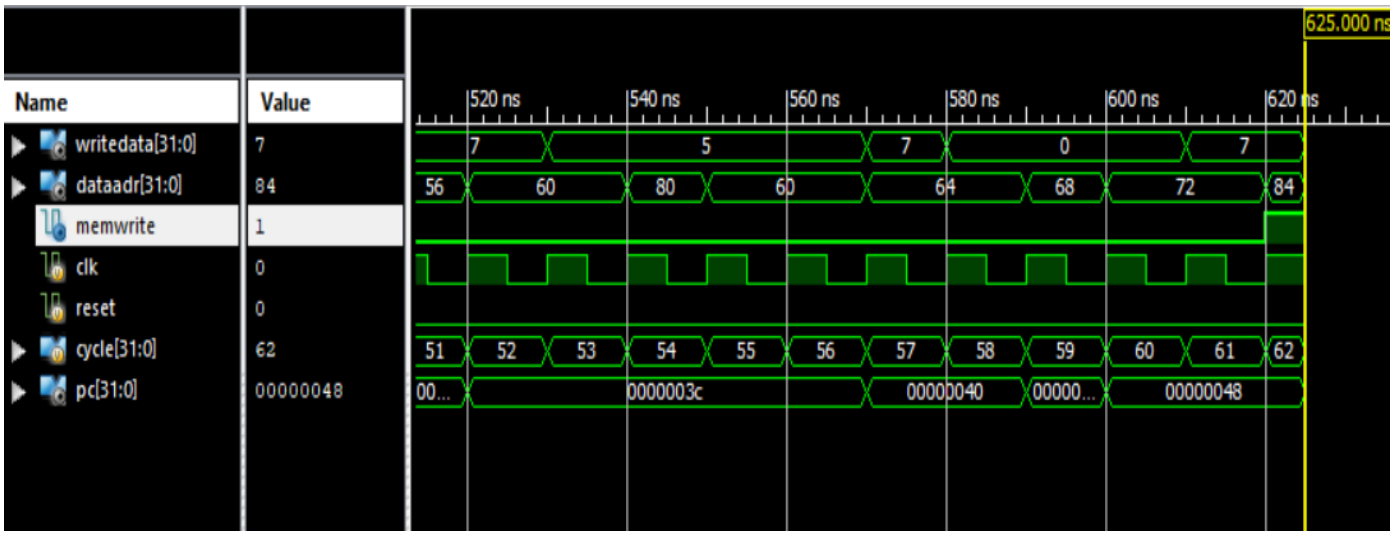


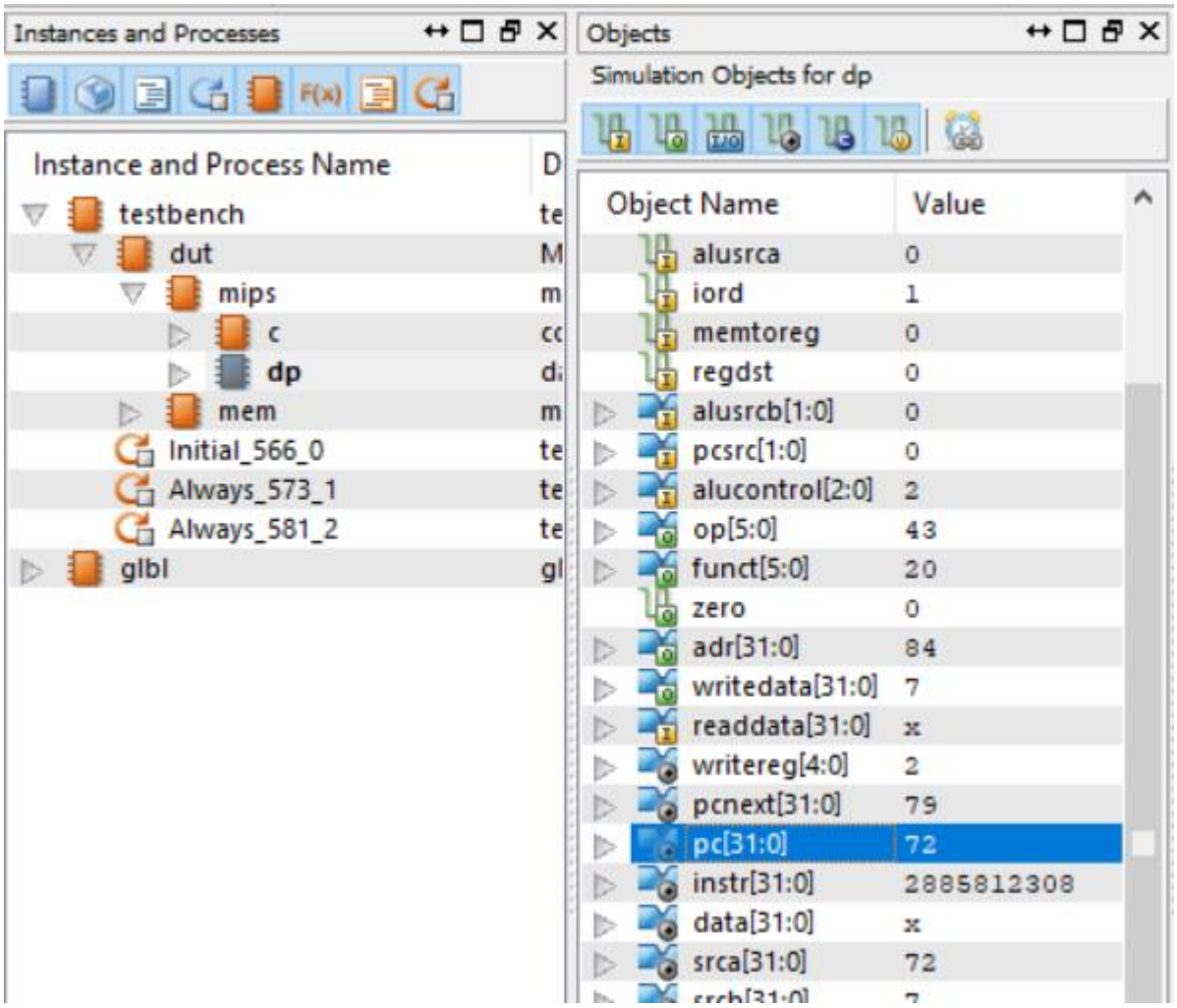
Important Note:

Once the project is created with the help of **multicycle.v** file it is necessary to have the **memfile.dat (//** memfile.dat is same for both single cycle and multi cycle programs**)** along with the multicycle.v file in the **same directory** as the project. Otherwise this may lead to errors or failure in execution.

Also for verifying the **addresses** it is necessary to add the program counter **PC** to the simulation by dragging it from the **dut/mips/dp/pc** part of the simulation window. It is important to observe that the values for program counter need to be in **Hexadecimal**, hence kindly change PC radix to hexadecimal and all other radix are signed decimal. Other this may lead to confusing outputs.

Simulation screen shots only include the last 150 to 200 ns as space is insufficient for whole simulation to be captured at one go.


625.000 ns
Name
Value
520 ns
540 ns
560 ns
580 ns
600 ns
620 ns
writedata[31:0] 7
dataadr[31:0] 84
memwrite 1
clk 0
reset 0
cycle[31:0] 62
pc[31:0] 00000048
7 5 7 0 7
56 60 80 60 64 68 72 84
51 52 53 54 55 56 57 58 59 60 61 62
00... 0000003c 00000040 00000... 00000048


Instances and Processes
Instance and Process Name
testbench
dut
mips
c
dp
mem
Initial_566_0
Always_573_1
Always_581_2
glbl
Objects
Simulation Objects for dp
Object Name
Value
alusrca 0
iord 1
memtoreg 0
regdst 0
alusrcb[1:0] 0
pcsrc[1:0] 0
alucontrol[2:0] 2
op[5:0] 43
funct[5:0] 20
zero 0
adr[31:0] 84
writedata[31:0] 7
readdata[31:0] x
writereg[4:0] 2
pcnext[31:0] 79
pc[31:0] 72
instr[31:0] 2885812308
data[31:0] x
srca[31:0] 72