1 – For Ahmed's Implementation, run test_lab4.v, For Matthew's implementation, run CPU.v and see Readme_Noyes.pdf

2 - To change the content of the instruction memory, please change the parameter filename in **Instruction_memory.v** module to any of the provided machine language files (All_inst.txt or test_fact.txt/factorial.txt). All_inst.txt tests all instructions, while test_fact.txt/factorial.txt tests fact (7).

3 – For Ahmed's implementation, set simulation time to 81 ps for each step. For Matthew's use 20 ns.

All_inst.txt content description:

0	001000 00111 00010 0001100000100000	Addi	$2 = $7 + 6176	$2 = 70 + 6176 = **6246** [#]
1	001100 00111 00011 0000000000001111	Andi	$3 = $7 & 15	$3 = 70 & 15 = **6**
2	001101 00111 00100 0000000000001111	Ori	$4 = $7 | 15	$4 = 70 | 15 = **79**
3	001010 00111 00101 0000000000101100	Slti	$5 = ($7 < 44)?1:0	$5 = (70 < 44)?1:**0**
4	001010 00011 00101 0000000000101100	Slti	$5 = ($3 < 44)?1:0	$5 = (**6** < 44)?**1**:0
5	001111 00000 00100 0000000000011111	Lui	$4 = 31	$4 = **31**
6	000000 00100 00001 00110 00000 100000	Add	$6 = $4 + $1	$6 = 31 + 10 = **41**
7	000000 00100 00001 00110 00000 100010	Sub	$6 = $4 - $1	$6 = 31 - 10 = **21**
8	000000 00100 00001 00111 00000 100100	And	$7 = $4 & $1	$7 = 31 & 10 = **10**
9	000000 00100 01000 00111 00000 100101	Or	$7 = $4 & $8	$7 = 31 | 80 = **95**
10	000000 00100 01000 01110 00000 100111	Nor	$14 = $4 nor $8	$7 = 31 nor 80 = **-96 (FFA0)$_{16}$**
11	000000 00100 01000 00111 00000 101010	Slt	$7 = ($4 < $8)?1:0	$7 = ($4 < $8)?**1**:0
12	000000 01000 00100 00111 00000 101010	Slt	$7 = ($8 < $4)?1:0	$7 = ($8 < $4)?1:**0**
13	100000 00001 01000 0000000000000010	Lb	$8 = Mem[$1+2] [*]	$8 = Mem[10+2] = (**2D00)$_{16}$**
14	100001 00001 01000 0000000000000010	Lh	$8 = Mem[$1+2]	$8 = Mem[10+2] = (**2D0E)$_{16}$**
15	101000 00001 01001 0000000000000010	Sb	Mem[$1+2] =Upper($9)	Mem[10+2] = Upper(90) = (**00)$_{16}$**
16	101001 00001 01001 0000000000000100	Sh	Mem[$1+4] = $9	Mem[10+4] = 90= (**005A)$_{16}$**
17	000000 01001 00001 00000 00000 011010	Div	P [**] = $9 / $1	
18	000000 00000 00000 01010 00000 010000	Mfhi	$10 = Upper(P)	$10 = 90%10 = **0**
19	000000 00000 00000 01011 00000 010010	Mfhl	$11 = Lower(P)	$11 = 90/11 = **9**
20	000000 01001 00010 00000 00000 011000	Mult	P = $9 * $2	P = 90*6246 = 562140 = $(893DC)_{16}$
21	000000 00000 00000 01010 00000 010000	Mfhi	$10 = Upper(P)	$10 = **8**
22	000000 00000 00000 01011 00000 010010	Mfhl	$11 = Lower(P)	$11 = **37852 = (93DC)$_{16}$**
23	000000 00000 01010 01100 00010 000000	Sll	$12 = $10 <<2	$12 = 8<<2 = **32**
24	000000 00000 01010 01101 00010 000010	Srl	$13 = $10 >> 2	$13 = 8>>2 = **2**
25	000010 00000000000000000000011011	J	Jump to 27	Jumps to (**INST 27**)
26	001000 00000 00001 0000000000000001	Addi	$1 = $0 + 1	(skipped)
27	000001 01110 00001 1111111111101100	Bgez	$14>0?go to PC-20*4	Will not branch
28 [+]	000001 01010 00001 1111111111101100	Bgez	$10>0?go to PC-20*4	Will branch **(INST 9)**

[#] The output of each instruction is shown in **bold**

[*] Memory is addressable from zero

[**] Special 32-bit register used to hold product and division result.

[+] **Beq**, **jal**, and **jr** instructions is already tested with the factorial implementation as shown in the report.